Commodore INFO

ONAFHANKELIJK BLAD VOOR COMMODORE GEBRUIKERS

Jaargang 6, NO. 5, juli/augustus 1989



LISTINGS

- Boekhouden C-16
- Databen C-64
- Kubus C-64
- Groeislang C-64
- Blocks Free C-64
- Zeeslag C-64
- Character Editor C-128
- Rubik's Clock Amiga

Zak McKracken
Digi-View 3.0
Teletext adapter
Geochart
Graphics op de 64
Raster interrupts
De Bootsector
C-128 OS

Vaste rubrieken

Geos Info
Basic Miniatuurtjes
Amiga DOS cursus

Commodore Info

Verschijnt 8x per jaar
Jaarg.6, no.5, aug. 1989

Uitgave:
Sala Communications
Uitgever:
Vic Sharfman
Redactie:
Ir. L. Sala hoofdredacteur
J. Bodzinga adj. hoofdred.
drs. J. Boers eindredacteur
drs. M. de Rooij &
J. Broekhuizen productie
drs. H. Zoete, H. Smeenk, drs. U. Schuurmans, R. Goudriaan, B. Munniksma, B. Venema, P. Boncz, MGCC/Johan & Johan

Redactiesecretariaat:
R. van Zalingen
Strip: Bert Tier
Illustraties: Ben van Mierlo

Advertentie-exploitatie:
Ing. V. Sala, Ing. B. Sala,
D. van Vlijmen
Weesperstraat 103
1018 VN Amsterdam
tel. 020-273198
Redactie adres:
Postbus 43048
1009 ZA Amsterdam
tel. 020-228871
Listingtelefoon:
(ma: 17.00-21.00) 02155-25162

Abonnementen en administratie:
Nicole Balke en Marjo Jansen
Postbus 43048
1009 ZA Amsterdam
tel. 020-248006
Vragen betreffende abonnementen ontvangen wij bij voorkeur schriftelijk, met meesturen van het omslagetiket.
Abonnement:
Voor 8 nummers *f* 47,50 of Bfr. 975 per jaar. Betaling op giro 1585491 (België: BBL nr. 310050602562) t.n.v. SAC/Commodore-Info. Oude nummers kunt U alleen krijgen bij vooruitbetaling van *f* 6,75 op de bovenstaande rekening. Ook telefonische opgave voor een abonnement is mogelijk. Bel GRATIS 06-02242222 (teleservice), elke dag tot 20.20 uur (dus ook in het weekend). België: 115555, dagelijks tot 22.00 uur. Deze telefoonnummers zijn alleen bedoeld voor opgave van NIEUWE abonnementen.
Opzegging dient schriftelijk te geschieden uiterlijk twee maanden voor de aanvang van een nieuwe abonnementsperiode van een jaar.

Omslagfoto: Audiomaster II (Aegis)

Zetwerk & druk: NDB, Zoeterwoude
Distributie:
In Nederland: Betapress, Gilze
In België: AMP, Brussel

ISSN: 0169-3085

Inhoud van dit nummer

Redactioneel

Terwijl de ene helft van Nederland ligt te sudderen aan de Costa del Sol en de andere helft zich schor schreeuwt bij de Tour is heel computerland in ruste. Het is duidelijk komkommertijd, en dat is goed te merken. Maar onderhuids maken velen zich alweer op voor het naderende seizoen. Her en der worden echter al naarstig voorbereidingen getroffen voor de grootste beurs in ons land: de Efficiency Beurs. Ongetwijfeld zal er eind september weer het nodige te bewonderen zijn in de RAI.
Hoe de homecomputer-markt zich zal houden onder de toenemende druk van PC's is de vraag. Er zal zeker een verdere verschuiving gaan plaatsvinden naar deze zwaardere machines. Maar daarnaast blijven er trouwe 64- en 128-gebruikers, en zal de Amiga , zeker op video- en muziekgebied, de strijd serieus kunnen blijven volgen. Ondertussen draait de pas ingestelde Hot Line van Commodore fantastisch. Gebruikers van elk soort Commodore machine kunnen met al hun vragen (en dat zijn er heel wat!) terecht bij een team van specialisten. Een goed initiatief waar menig concurrent een voorbeeld aan zou kunnen nemen.

Machiel de Rooij

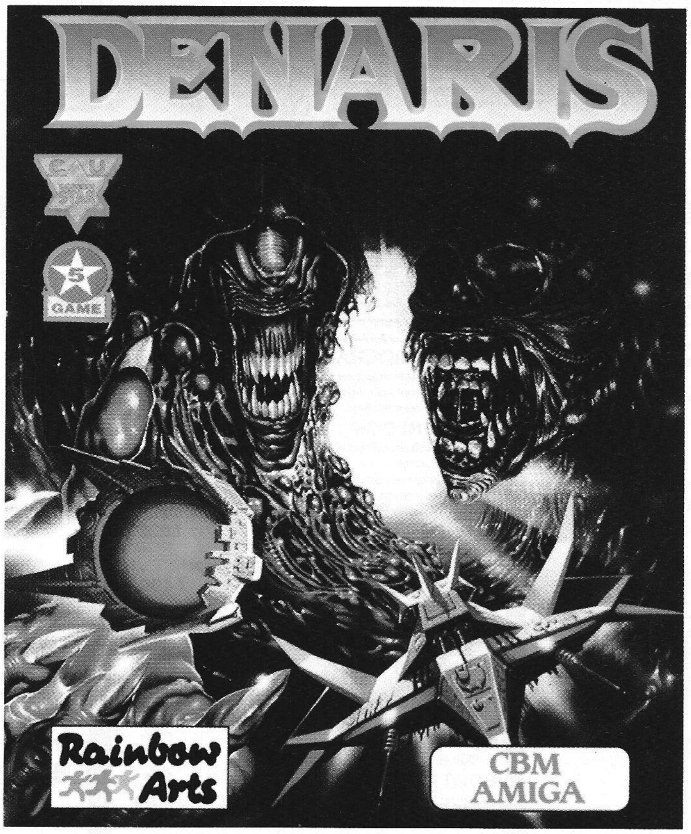
De zomerse stilte aan het softwarefront wordt nu en dan verbroken door goede computerspel-ontwerpen. Skweek en Zak MacKracken scoren hoog op de spellijst van 1989. Piraten gooien echter volgens software-importeurs roet in het Hollandse computerspel-eten.

Zomerkoninkjes

Er heerst ook deze zomer de gebruikelijke rust aan het Nederlandse computerspellenfront. De ontwerpers hebben natuurlijk ook recht op vakantie. Even uitrusten van de enorme hoeveelheden geslaagde en nog grotere hoeveelheden minder geslaagde spellen die dankzij hun inspanningen op de markt verschijnen. Nu is het tijdens de zomermaanden altijd wat rustiger in het hele computergedoe, maar enkele reorganisaties in de Engelse 'spelletjesfabrieken' hebben vermoedelijk ook iets met de wat vroeg ingevallen spelrust in Nederland te maken. Een andere oorzaak van de kalmte is het 'piratencircuit'. Een grote spelimporteur in Nederland maakt zich met regelmatige tussenpozen nogal boos op de kopieermentaliteit van de Nederlandse computerbezitter. Er is zelfs gedreigd om het spelaanbod voor de Ataricomputers sterk in te krimpen. Of dat ook gevolgen heeft voor het aanbod van Commodore- en Amiga-spellen heeft is moeilijk te overzien. Laten we maar eens gaan kijken naar de laatste aanbiedingen waaronder een paar spellen zeker de moeite waard zijn.

Denaris

Onze speltester is redelijk tevreden over het spel Denaris van Rainbow Arts (een uitgave van US Gold). Een behoorlijk spel waarbij de speler zijn handen vol heeft aan het afslaan van de massale aanvallen van de tegenstander. Onze (dus aan de goede kant knokkende) vechtmachine moet zich door een aantal velden worstelen waarbij aan het eind van ieder veld een ongelooflijk sterke tegenstander moet worden overwonnen. Ons vaartuigje blijkt in staat om zich tijdens de ruimteoorlog steeds beter te bewapenen. Goed schietspel met uitstekende bewegingen en plaatjes op het beeldscherm. Denaris is van Rainbow Arts en kost voor de Amiga 500 *f* 75,-.



Denaris

Skweek

We ontvingen kort geleden een pakje uit Engeland waaruit, na erop te duwen, vreemde geluiden kwamen. Het pakje blijkt, na heel voorzichtig openmaken, een afgrijselijk lelijk rubber babyspeelgoedje te bevatten. De vorm van het wezentje zal bij iedere baby spontaan de stuipjes verdubbelen! Het vreemdvormige wezentje is volgens de bijgesloten brief de originele uitvoering van de hoofdrolspeler in het nieuwe computerspel 'Skweek!'. Het is een computerspelletje van Loriciels en wordt uitgebracht door US Gold. Skweek is een razendsnel puzzelspel voor verschillende computers. Het is te spelen met twee spelers. De opdracht is eenvoudig. Maak met Skweek alle blauwe velden die je op het beeldscherm ziet roze door er overheen te lopen. Na iedere voltooide opdracht wisselt het speelveld en wordt het uiteraard moeilijker. De hoofdrolspeler in het spel lijkt inderdaad redelijk op het eerder toegezonden speelgoedje. De talrijke tegenstanders zijn eveneens mooi uit-

gevoerde graphics en de speelvelden zien er goed uit. Overigens is het verstandig om de computer na de automatische start even zijn gang te laten gaan. Onder begeleiding van een uitstekend muziekje laat Skweek zelf even zien wat er allemaal mogelijk is in dit spel. Zo maak je kennis met een aantal tegenstanders en zie je vooraf de afbeeldingen waarmee je bonuspunten kunt verzamelen. Aan het begin van het spel beschik je over negen levens, welk aantal je tijdens het spel nog kunt aanvullen door zogenaamde 'bonusbeertjes' te verzamelen. Het spel Skweek is weliswaar niet nieuw in opzet, maar zit vol met grapjes en onverwachte situaties. Een prachtig, zeer gevarieerd, verslavend en redelijk moeilijk spel voor alle leeftijden. Toen we het schijfje kregen, was er nog geen omslag, maar we hebben inmiddels wel de prijs (*f* 79,50) voor de Nederlandse uitvoering doorgekregen. Skweek ligt vermoedelijk deze maand in de computerwinkel.

Battlehawks 1942

Deze uitgave van Lucasfilm Games is iets voor de liefhebbers van een kruising tussen een vliegsimulator en een schietspel. Weer een prima spel van Lucasfilm. Zelfs de nogal ervaren (computerspel)vliegers van de redactie krijgen er lol in! Snel klimmend op je doel af en knallen maar! In steile bochten blijf je je tegenstander schietend achtervolgen, totdat er een waarschuwing komt dat je je daling of klim iets te drastisch hebt ingezet! Eerst weer klimmen of 'stall' (gevaarlijk snelheidsverlies tijdens de klim) opheffen, voor het schieten verder gaat! Met een uitstekende (Engelse) handleiding en boekje over de historische 'helden' in hun gevechtsvliegtuigen voel je je met dit spel voor enige tijd een echte hemelbestormer. Via het boekje kan een keuze gemaakt worden uit verschillende historische luchtgevechten. Uitleg over afstand tot de vijand, hoeveel vijandige vliegtuigen aan de strijd deelnemen enz. Het uit twee schijven bestaande spel kost voor de Amiga computers *f* 99,-.

Danger Freak

Danger Freak van Rainbow Arts, uitgebracht door US Gold, is een stuntman-spel. Ook hierbij ontbreekt nog de verpakking, zodat we geen plaatje kunnen laten zien. In Danger Freak gaat het om een serie in ontwerp sterk op elkaar lijkende behendigheidsspellen. We zijn deze spelontwerpen al eerder in oudere spelversies tegen



Battlehawhs 1942

gekomen. bijvoorbeeld in spellen als 'Hang On' van Sega of 'ExciteBike' van Nintendo. Op een motorfiets een baantje trekken en intussen obstakels vermijden. Een andere spelletje op de zelfde schijf is aardiger en moeilijker. Op een motorrace-circuit mag je als vierde coureur meerijden om de zege. Het duurt even voor je door hebt hoe je moet sturen. Het onderdeel met de waterscooter is weer een variatie op het voorafgaande. De begeleidende muziek is goed en de grafics mogen er ook zijn. Verder is het een niet al te interessant spelschijfje voor de Amiga en C64/128, voor *f* 79,50 en zal in juli verschijnen.

Human Killing

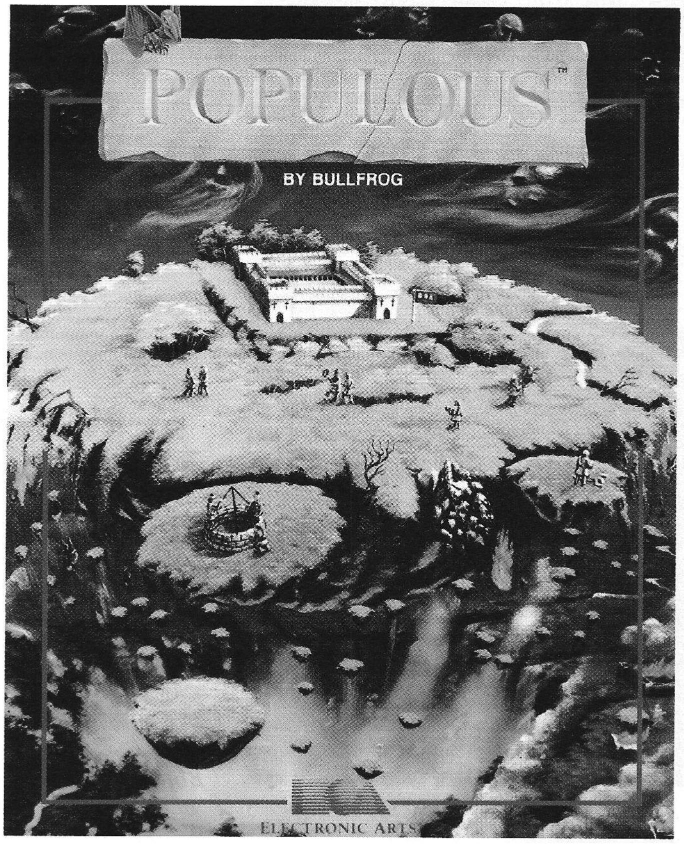
Veel minder te spreken zijn we over HKM van US Gold. Dit spel is een zogenaamde vechtsportsimulatie. Springen en bukken en schoppen enz.... Met op de achtergrond een paar aardige plaatjes is het verder een saaie bedoening! Verkijk je niet op het Rambo-achtige en daardoor aantrekkelijk aandoende omslagje? Kijk maar even verder in de computerwinkel voordat je je geld aan dit spel uitgeeft. HKM (Human Killing Machine) kost *f* 89,50.

Diep

Ook al niet om te juichen is The Deep van Cream. Aardige plaatjes, maar verder.....een spelontwerp uit de beginjaren van de computer ondanks een poging om de zaak wat aan te kleden met redelijk goede en mooi bewegende figuurtjes als tegenstander. The Deep voor Amiga kost *f* 89,50.

Trollen

Veel beter is het zogeheten etagespel Realm of the Trolls van Rainbow Arts. Veld na veld, ladder op en ladder af, moeten de aanwezige goudstukken en andere waardevolle zaken worden opgepakt. Dat dat niet zomaar gaat, zal iedereen begrijpen. Trollerige tegenstanders proberen je dwars te zitten. Lukt het de trollen niet dan zijn er nog de valstrikken en -kuilen die de weg naar het succes vrijwel onbegaanbaar maken. Realm of the Trolls is een aardig computerspel voor de Amiga en kost *f* 89,50.



Populous

Populous van Bullfrog. Om je vingers bij af te likken! Dit spel dat in Nederland door Homesoft Benelux op de markt wordt gebracht is van ongekende klasse. "Hoor mij nou..." Er komt wel eens het verwijt dat de speltester volstrekt onvoldoende uit zijn bol gaat! Goed...! Inbinden dus... Populous is waanzinnig! In de eerste plaats is het spel enigszins moeilijk te doorgronden. Ho! laat je nu niet onmiddellijk afschrikken! In de ontwikkeling van dit spel is werk verzet. En niet alleen ter lering en de vermaak van de ontwerpers! Als je eenmaal in de gaten hebt hoe alles werkt ben je verslaafd aan een reuzespel!

Waar gaat het allemaal om? Geen lange inleiding. Gewoon een groep 'slechten' en een groep 'goeden' (wij natuurlijk), die kans moeten zien de heerschappij over een prachtig in 'perspectief-beeldscherm' gebracht gebied te krijgen. Zoals bij ieder spel ben jij de hopelijk succesvolle leider. Omdat je niet rechtstreeks je 'volgelingen' kunt besturen is het noodzakelijk enige listige trucjes te bedenken. Lees de handleiding op je gemak door en geniet intussen van de werkelijk schitterende intro! Populous is wat geluid en grafische plaatjes betreft een hoogstaand computerspel voor de Amiga. De prijs is (zoals wel vaker) niet mis (*f* 89,50) maar het spel zal de computerspeler wel lange tijd aan de monitor kluisteren.



Zak McKracken

Nog een aardig spel dat de spelfanaat onder ogen kreeg is Zak McKracken van Lucasfilm Games. Een soort adventure met humoristische situaties. Een adventure schrikt veel spelletjesliefhebbers af maar dit is een spel waar ook de tegenstanders van deze spelvorm genoegen aan kunnen beleven. De verschillende opdrachten hoeven alleen maar met de muis te worden 'aangeklikt'. Door de looprichting aan te wijzen kun je de hoofdfiguur verplaatsen. Zak McKracken is journalist bij een niet al te groot krantje. In de verpakking is een exemplaar van deze krant bijgesloten. Hij wil nu eindelijk wel eens een goed verhaal maken. De hoofdredacteur stuurt hem echter naar weer een vervelende opdracht. "Dubbelkoppige eekhoorn valt kampeerders aan". Zak vertrekt met een vliegticket vanuit zijn kamer om ons vervolgens de rest te laten opknappen! Een spel vol grappen en grollen. Niet eenvoudig, maar er staan voldoende aanwijzingen in de handleiding (Engels) en de bijgeleverde krant (uiteraard ook Engelstalig) om de speler op weg te helpen. Om vast wat te verklappen: aan zijn opdracht komt McKracken nauwelijks toe! Na een invasie van buitenaardse wezens gaat alles fout. De vreemdelingen plaatsen een machine op aarde die het IQ van iedereen in snel tempo doet dalen. Er is maar een persoon die een einde kan maken aan die dreiging...Zak McKracken....en dat ben jij! De prijs waarvoor je dit kunt proberen is *f* 89,50

Het vorige deel van de cursus Graphics op de Commodore 64 ging over sprites. In dit derde deel van deze cursus gaan Michel de Boer en Hylke Sprangers verder in op het onderwerp sprites. Was de vorig aflevering misschien oude koek voor de wat verder gevorderde programmeur, deze aflevering bevat zeker genoeg informatie voor iedereen om mee aan de slag te kunnen. Mocht u trouwens vragen hebben of graag zien dat op een bepaald onderwerp wat dieper wordt ingegaan, dan kunt u altijd een brief sturen.

Graphics op de 64

Deel 3: sprites vervolg

In de vorige aflevering zijn we begonnen met de wondere wereld van de sprites. Daarin is onder andere uitgelegd wat sprites zijn en hoe ze betrekkelijk eenvoudig te maken zijn. In deze aflevering zullen we eerst beginnen met een korte herhaling, zodat u weer weet hoe alles in elkaar zit. Daarna zullen we de verschillende onderdelen behandelen, die de sprites tot de meest fascinerende graphische objecten maken op de Commodore. Deze onderdelen zijn onmisbaar voor de echte spelprogrammeur, omdat sprites vaak de pijlers van snelle, flitsende video games vormen. De onderdelen waar we het nu over hebben zijn de volgende: * Multi color sprites * Overlappen van sprites * Botsingen van sprites * Meer dan 100 kleuren per sprite * Applikatie programma's

0	0	1	1	1	1	0	0
0	1	0	0	0	0	1	0
0	1	0	0	0	0	1	0
0	1	0	0	0	0	1	0
0	0	1	0	0	1	0	0
0	0	0	1	1	0	0	0
0	1	1	0	0	1	1	0
0	0	0	0	0	0	0	0

Sprite adressen

Adres	Functie
53248	X-coordinaat sprite 0
53249	Y-coordinaat sprite 0
53250	X-coordinaat sprite 1
53251	Y-coordinaat sprite 1
53252	X-coordinaat sprite 2
53253	Y-coordinaat sprite 2
53254	X-coordinaat sprite 3
53255	Y-coordinaat sprite 3
53256	X-coordinaat sprite 4
53257	Y-coordinaat sprite 4
53258	X-coordinaat sprite 5
53259	Y-coordinaat sprite 5
53660	X-coordinaat sprite 6
53261	Y-coordinaat sprite 6
53262	X-coordinaat sprite 7
53263	Y-coordinaat sprite 7
53264	MSB register
53269	Sprite enable register
53271	Vertikale vergroting
53277	Horizontale vergroting
53276	Multi color register
53285	Kleur met bitcombinatie 01
53286	Kleur met bitcombinatie 11
53287-53294	Kleur registers sprite 0 tot en met sprite 7 In multi color mode bevatten deze registers de kleur met bitcombinatie 10
2040-2047	Sprite pointers sprite 0 tot en met sprite 7
53275	Sprite-karakter prioriteit
53278	Sprite-sprite botsing
53279	Sprite-karakter botsing

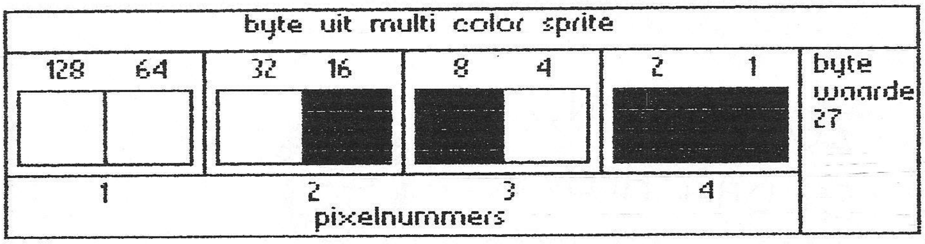
Figuur 1
Sprite adressen

Herhaling

Als geheugensteun zullen we in deze paragraaf het maken van een sprite in vogelvlucht behandelen. Een sprite is een beweegbaar figuurtje dat op elke willekeurige plaats van het scherm kan worden gezet. Zo'n sprite is opgebouwd uit een raster van 504 pixels, te weten 24 pixels in horizontale en 21 pixels in de vertikale richting. Acht horizontale pixels vormen samen een byte. Een horizontale rij van 24 pixels bestaat zo uit 3 bytes. De hele sprite bestaat dus uit 21 * 3 = 63 bytes. Om een sprite te maken, moet eerst een ontwerp in een raster tekenen en daarna dit raster omrekenen in 63 bytewaarden. Zo'n bytewaarde wordt verkregen door de waarden van de bits die aanstaan bij elkaar op te tellen. De waarde van een bit is 2 ^ bitnummer, waarbij de bits van rechts naar links genummerd zijn, beginnend bij 0. Deze bytewaarden moeten dan worden opgeslagen in het geheugen tussen de adressen 0 en 16384 met uitzondering van het geheugengebied tussen 4096 tot 8192. Daarbij moet de eerste bytewaarde op een adres staan, dat een veelvoud is van 64. De veelvoud van 64 is nodig, omdat de computer moet worden verteld waar de betreffende sprite in het geheugen staat. Dit moet gedaan worden met een zogenaamde sprite pointer. De waarde van de sprite pointer wordt berekend door het beginadres van het geheugengebied waarin de sprite staat te delen door 64. Om de sprite op het beeldscherm te zetten moeten de x- en y-coordinaten van de sprite aan de computer worden doorgegeven. Als laatste moet het juiste bit in het sprite enable register worden aangezet. Zie voor alle benodigde sprite adressen de tabel in figuur 1. Tot zover de herhaling van de vorige aflevering.

Multi color sprites

Tot nu toe hebben we alleen nog maar sprites behandeld, die een kleur hebben; de zogenaamde single color sprites. Er bestaat voor de sprites ook een multi color mode, net zoals bij karakters. In deze mode is het mogelijk om een sprite drie kleuren te geven. Dit gaat echter wel ten koste van de resolutie van de sprite. Want de sprite bestaat dan nog maar uit een raster van 12 bij 21 pixels in plaats van 24 bij 21 pixels. Een pixel in een multi color sprite is twee keer zo breed als een pixel in een single color sprite. De sprite wordt dus groffer. De pixels worden nu door twee bits in plaats



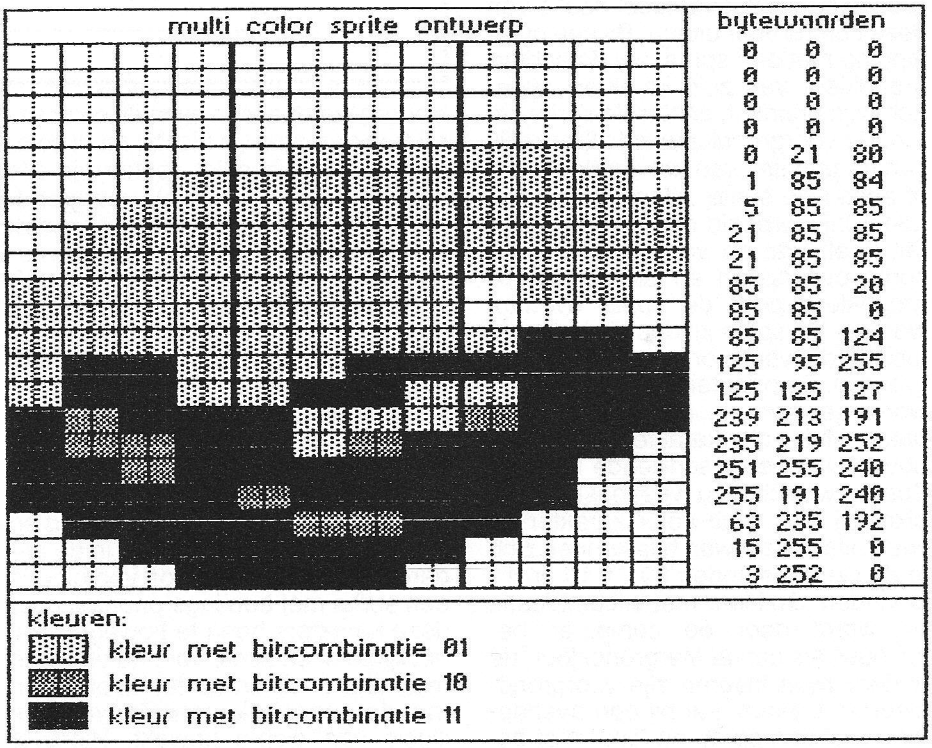
Figuur 2

van een bit gerepresenteerd. Deze bits bepalen of het pixel aan of uit staat en ook de kleur van het pixel, als deze aan staat. Met deze twee bits kunnen vier combinaties van bitstanden worden gemaakt (00 01 10 11). De combinatie 00 betekent dat het pixel uitstaat. De overige drie combinaties staan voor de drie kleuren. (het pixel staat dan aan). Een multi color sprite bestaat net als een single color sprite uit 63 bytes. Alleen correspondeert een byte nu maar met vier pixels; twee bits per pixel. Deze twee pixels liggen steeds naast elkaar in het byte. Bit 0 en 1 corresponderen met het meest rechtse pixel en bit 6 en 7 met het meest linkse. Helaas is er een beperking bij het kiezen van de kleuren. U kunt namelijk niet elke sprite drie willekeurige kleuren geven. Twee van de drie kleuren moet u voor alle sprites gezamelijk kiezen. De derde kleur kan, net zoals in de single color mode, voor elke sprite afzonderlijk worden gekozen. De eerste en de tweede kleur moeten respectievelijk in de adressen 53285 en 53286 worden gepoked. Deze twee kleuren corresponderen met de bitcombinaties 01 en 11. Zet u bijvoorbeeld de bitcombinatie 11 in bit 3 en 2 van een byte, dan heeft dat tot gevolg dat het tweede pixel van rechts aan staat met de kleur die in adres 53286 staat. De derde kleur (bitcombinatie 10) moet gewoon in het kleurregister van de sprite worden gepoked. Ter verduidelijking zullen we hier een voorbeeld geven (zie figuur 2). Dit figuur laat een byte van een multi color sprite zien. Het eerste pixel staat hier uit. Het tweede pixel heeft de kleur die in adres 53285 (bitcombinatie 01) staat. Het derde pixel heeft de kleur die in het kleur register (bitcombinatie 10) staat en het vierde pixel heeft de kleur die in adres 53286 (bitcombinatie 11) staat. We moeten nu alleen nog weten hoe de computer in de multi color mode wordt gezet. Dit kan voor elke sprite afzonderlijk gedaan worden. Hiervoor is een apart sprite multi color register beschikbaar. In dit register is er voor elk van de sprites een bit dat aangeeft of de sprite in single of multi color mode staat. Als het bit van een sprite uit staat is de sprite single color en als het bit aan staat multi color. Het sprite multi color register is adres 53276. In figuur 3 staat een hele multi color sprite uitgewerkt met bijbehorende bytewaarden. Het onderstaande programma zet deze sprite vijf keer op het scherm.

```
10 rem multi color sprite
20 poke 53280,0:poke 53281,0
30 print chr$(147)
40 for i=0 to 62:read a
50 poke 704+i,a:next
60 for i=0 to 8 step 2:reada
70 poke 53248+i,a:read a:
   poke 53249+i,a
80 next
90 poke 53285,1:poke 53286,14
100 for i=0 to 4:
   poke 53287+i,11
110 poke 2040+i,11:next
120 poke 53276,31
130 poke 53271,4:poke 53277,4
140 poke 53269,31
190 rem bytewaarden
200 data 0,0,0,0,0,0
210 data 0,0,0,0,0,0
220 data 0,21,80,1,85,84
230 data 5,85,85,21,85,85
240 data 21,85,85,85,85,20
250 data 85,85,0,85,85,124
260 data 125,95,255,125,125,
    127
270 data 239,213,191,235,219,
    252
280 data 251,255,240,255,191,
    240
290 data 63,235,192,15,255,0
300 data 3,252,0
400 rem coordinaten
410 data 100,80,240,80,160,
    130, 100,200
420 data 240,200
```

Overlappen van sprites onderling

Als sprites vlak bij elkaar op het scherm staan, zullen gedeelten van de sprites overlappen. Welke sprite zal nu welke sprite overlappen? Het overlappen van sprites is aan vaste regels gebonden. Elke sprite heeft een eigen overlappings prioriteit. Als twee sprites vlak bij elkaar in de buurt staan, zal de sprite met de hoogste prioriteit de ander overlappen. De overlappingsprioriteiten van de sprites zijn als volgt geregeld: de sprite met het laagste spritenummer heeft de hoogste prioriteit. Als bijvoorbeeld sprite 2 en sprite 5 bij elkaar in de buurt staan, dan zal sprite 2 geheel zichtbaar zijn en sprite 5 gedeeltelijk. De rest van sprite 5 zit achter sprite 2. Overlappen van sprites met karakter



Figuur 3

Sprites en karakters kunnen tegelijkertijd op het scherm worden gezet. Net zoals bij de sprites onderling zal dus overlapping optreden als een sprite en een karakter gedeeltelijk of geheel op dezelfde plaats op het scherm staan. Ook nu is de overlapping weer geregeld met een overlappingsprioriteit. Als het karakter de hoogste prioriteit heeft, zal deze de sprite overlappen; anders zal de sprite het karakter overlappen. Wie de hoogste prioriteit heeft, wordt nu niet door de computer geregeld, maar moet door de programmeur worden bepaald. Hiervoor is een speciaal sprite-karakter prioriteiten register. Dit register bevindt zich op adres 53275. In dit register correspondeert elk bit weer met een spritenummer. Bitnummer 0 komt overeen met spritenummer 0. Als een bit uit staat, dan heeft de overeenkomstige sprite de hoogste prioriteit. Als het bit aan staat, heeft het karakter de hoogste prioriteit. Voor single color karakters is bovenstaande beschrijving van overlappingen compleet. De overlappingen van sprites met multi color karakers en karakters met verschillende achtergrondkleuren behoeven nog verdere uitleg, omdat de computer bij deze overlappingen onderscheid maakt tussen de verschillende kleuren van de karakters. Karakters met verschillende achtergrondkleuren hebben een voorgrondkleur (net zoals single color karakters) en een eigen achtergrondkleur. (Zie eventueel deel 1 van deze cursus voor uitleg). Bij een overlapping met een sprite, zal de achtergrondkleur van zo'n karakter, ongeacht zijn prioriteit, achter de sprite zitten. De voorgrondkleur zal afhankelijk van de prioriteit van het karakter voor of achter de sprite zitten. Als het karakter bijvoorbeeld de hoogste prioriteit heeft, zal de voorgrondkleur de sprite overlappen en zal de achtergrondkleur door de sprite overlapt worden. De sprite zit dus als het ware tussen de voorgrond en de achtergrond in. De prioriteit van het karakter wordt gewoon via adres 53275 bepaald. Multi color karakters zijn opgebouwd uit drie verschillende kleuren. Zoals u wellicht nog weet bestaat een pixel bij een multi color karakter uit twee bits. Deze twee bits kunnen zich in de vier toestanden 00 01 10 en 11 bevinden. De kleur met bitcombinatie 01 wordt door de computer beschouwt als een achtergrondkleur; de andere twee kleuren zijn voorgrondkleuren. Daarom zal bij een overlapping met een sprite de kleur met bitcombinatie 01 altijd door de sprite overlapt worden. Het onderscheid in kleuren wordt alleen bij multi color karakters gemaakt en niet bij multi color sprites. Bij multi color sprites zijn alle kleuren (van de sprite) voorgrondkleuren en zullen allemaal afhankelijk van de prioriteit voor of achter een karakter zitten. Misschien vraagt u zich nu af waarom 1 kleur altijd door een sprite overlapt wordt, en waarom niet alle kleuren, afhankelijk van de prioriteit, voor of achter de sprite zitten. Doordat de overlapping op deze manier is geregeld, is het mogelijk om 3D animaties te maken. Hiervoor moeten een aantal objecten in de kleur met bitcombinatie 01 worden getekend en een aantal andere karakters in de andere twee kleuren. Door vervolgens een sprite met een lage prioriteit over deze karakters heen te bewegen, zal het lijken of de sprite voor de objecten met kleur 01 zit en achter de objecten met de andere kleuren. Zo wordt de illusie van diepte gewekt. Hierover zullen we in een aflevering over animatie meer vertellen.



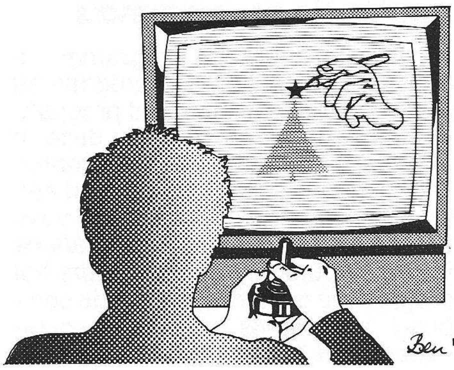
Botsingen tussen sprites

U zult mischien inmiddels wel begrijpen waarom sprites zulke mooie grafische objecten zijn. U kunt er vanuit Basic maar liefst acht tegelijk onafhankelijk over het scherm laten bewegen. Vanuit Machinetaal kunnen er zelfs meer dan acht sprites op het scherm worden gezet, maar daar vertellen we over een paar afleveringen meer over. Het feit dat sprites bij de meeste video games de pijlers van het programma en de actie vormen, vloeit onder meer voort uit de flexibiliteit van de sprites, oftewel het makkelijk kunnen animeren van de sprites. In de vorige paragraaf hebben we gezien dat sprites onderling en sprites met karakters, kunnen overlappen. De regels van het overlappen tussen sprites onderling zijn heel simpel. Nu zou het mooi zijn als je ook te weten zou kunnen komen of er werkelijk een overlapping van twee sprites plaats vindt. Met andere woorden, of er een

botsing tussen twee of meerdere sprites plaats vindt. Denkt u maar eens aan een schietspelletje. Als bijvoorbeeld bij Galaxions een indringer is geraakt met een kogel, moet het computerprogramma dit wel weten, om vervolgens de indringer te elimineren. Wel, het botsen van sprites houdt de Commodore 64 ook bij, en wel in het geheugenadres 53278. Voor elk van de acht standaard sprites is in dit adres een bit gereserveerd; bit 0 voor sprite 0 en bit 7 voor sprite 7. Normaal staat voor elke sprite het corresponderende bit op nul. Als tussen twee sprites een botsing wordt gedetecteerd door de computer, worden de bits van beide sprites op een gezet, ten teken dat beide sprites botsen. Twee sprites zijn alleen in botsing als ten minste een pixel van de ene sprite precies op de zelfde plaats van het scherm staat als een pixel van de andere sprite. Met een pixel bedoelen we in dit verband een rasterpunt dat aan staat. Zoals gezegd wordt het bit van de betreffende sprite in adres 53278 aan gezet als die sprite botst. Dit bit blijft aan, samen met de andere bits in adres 53278 die aan staan, totdat dit register is uitgelezen met PEEK. Na een PEEK(53278) wordt adres 53278 weer helemaal op nul gezet. Zo kunt u elke botsing detecteren, hoe kort ze ook zijn, want de bits in adres 53278 worden pas weer op nul gezet, nadat het register is uitgelezen. Dit houdt wel in dat als het register eenmaal uitgelezen is, en dus gereset, u de inhoud van het register kwijt bent, tenzij u de inhoud van het register in een variabele stopt. Bij het botsen van twee sprites maakt het niet uit of een van beide of eventueel beide sprites in de multi color mode staan. Het botsen van een multi color sprite gaat precies het zelfde als het botsen van een single color sprite. U moet wel uitkijken met de kleuren van een sprite. U kunt bijvoorbeeld de kleur van de sprite gelijk maken aan de achtergrond kleur van het scherm. U ziet de sprite dan niet, maar voor de computer staat er gewoon een sprite op het scherm en zal dan ook zonodig een botsing van deze sprite registreren.

Sprites en karakters

Naast het botsen van sprites onderling, houdt de Commodore 64 ook het botsen van sprites met karakters bij. Dit kan heel handig zijn. Denk bijvoorbeeld aan het geval dat u als sprite een vlieg- tuigje hebt, dat door de lucht vliegt. U zou dan bijvoorbeeld als achtergrond wat wolken kunnen maken van karakters. Als de computer vervolgens een botsing detecteert tussen de sprite en een karakter, hier het vliegtuigje en de wolk, kunt u bijvoorbeeld het vliegtuigje laten crashen. Wat u overigens ook kunt doen, is de overlappingsprioriteit van het vliegtuigje lager maken dan de karakters, waardoor het lijkt of het vliegtuigje achter cq. door de wolk heen vliegt. Maar we haden het over het botsen van sprites met karakters. De computer houdt in adres 53279 voor elk van de acht sprites het botsen met karakters bij. Adres 53279 is equivalent met adres 53278, met het verschil dat het eerste adres botsingen tussen sprites en karakters, en het tweede adres botsingen tussen sprites onderling registreert. In adres 53279 is voor elke sprite een bit dat aan wordt gezet als de computer een botsing registreert. Ook adres 53279 wordt gereset als het wordt uitgelezen. Als u bijvoorbeeld wilt weten of sprite 1 met een karakter in aanvaring is, kunt dat als volgt berekenen. IF (PEEK (53279) AND 2) THEN REM BOTSING In de IF conditie wordt gekeken of het tweede bit van adres 53279 aan staat. Als dat zo is, levert PEEK (53279) AND 2 een getal op groter dan nul, namelijk 2, en wordt het gedeelte na de THEN uitgevoerd.



Staat het tweede bit echter uit, dan levert de boolean expressie na de IF 0 op en wordt het THEN gedeelte van het statement niet uitgevoerd. Dit komt omdat de Commodore 64 het getal 0 als FALSE en elk ander getal als TRUE beschouwt. Bij het botsen tussen twee sprites onderling hebben we opgemerkt dat het niet uit maakte of de betreffende sprites single of multi color sprites waren. Bij het botsen tussen karakters en sprites ligt dit echter anders. Het botsen van een sprite tegen een single color karakter verloopt normaal, maar bij het botsen van een sprite tegen een multi color karakter moet u oppassen. Bij het detecteren van een botsing van een sprite tegen een multi color karakter beschouwt de Commodore 64 alleen de bitcombinaties 10 en 11 als een pixel dat aanstaat. Als een sprite dus een aanvaring heeft met een pixel 01 van een multi color karakter wordt er dus geen botsing geregistreerd. Het volgende programma geeft een voorbeeld van wat er zoal met de botsing registers gedaan kan worden. U moet een multi-color racewagen met de joystick over een kronkelweg besturen. Met adres 53279 wordt getest of de racewagen niet van de weg afraakt.

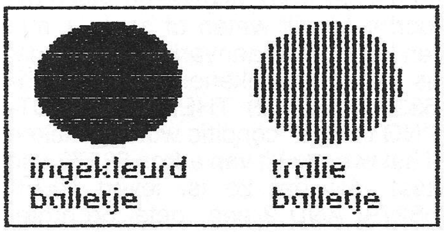
```
10 rem
20 rem auto race
30 rem
40 print chr$(147):restore
50 for x=0 to 62:read a:poke
   832+x,a
60 next:poke 2040,13:for x=0
   to 7
70 read i,a:poke
   53270+i,a:next
80 poke 53269,1:xc=175:sc=0
90 a=peek(53279):a=15:poke
   53248,xc
100 poke 53249,100:for x=0 to
    26
110 print
    tab(15)"#"spc(9)"#":next
120 b=int(rnd(0)*3)-1
130 if a+b<26 and a+b>3 then
    a=a+b
140 print
    tab(a)"#"spc(9)"#":sc=sc+1
150 if peek(53279) then 190
160 q=peek(56320):if q=119
    then xc=xc+2
170 if q=123 then xc=xc-2
180 poke 53248,xc:goto 120
190 for x=1024 to 1104:poke
    x,32:next
200 if sc>hs then hs=sc
210 print chr$(19)"score:"sc
220 print "high-score:"hs
230 wait 197,63:goto 40
240 data 12,192,0,12,192,0,
    204, 204,0
250 data 234,172,0,229,108,0,
    2 34,172,0
260 data 229,108,0,42,160,0,
    38,96,0
270 data 38,96,0,38,96,0,42,
    160,0
280 data 201,140,0,201,140,0,
    233,172,0
290 data 202,140,0,194,12,0,
    2, 0,0
300 data 0,0,0,0,0,0,0,0,0,
    1,1, 6,1
310 data 7,1,10,0,11,0,15,7,
    16,2, 17,6
```

Mengkleuren

In de vorige aflevering hebben we beloofd om sprites in meer dan 100 kleuren te maken. De hardware van de Commodore 64 kan slechts 16 verschillende kleuren genereren. Met slimme software trucs kan het kleu-

renscala worden uitgebreid tot maar liefst 136 kleuren. Hieronder zullen we uitleggen hoe dit gedaan kan worden. Als u wel eens met kleuren geexperimenteerd heeft, zult u wel gemerkt hebben dat bij bepaalde kleurencombinaties vreemde effecten optreden, zoals het in elkaar overvloeien van verschillende kleuren. Meestal zijn deze effecten ongewenst en vervelend, deze keer kunnen we er echter handig gebruik van maken. Het principe voor het uitbreiden van het kleurenscala berust namelijk op het mengen van verschillende kleuren; op dezelfde manier maakt een schilder nieuwe kleuren uit de kleuren die hij al heeft. Door bijvoorbeeld blauw en geel te mengen, wordt groen verkregen. Zo kunnen ook kleuren van sprites gemengd worden. Als u twee sprites met verschillende kleuren op dezelfde plaats op het scherm zet, zal er een voor de ander staan en zullen de kleuren niet gemengd worden. Als u echter van de voorste sprite een tralie maakt, dan zal een gedeelte van de achterste sprite door de voorste sprite heenschijnen. Doordat de twee verschillende kleuren nu heel dicht bij elkaar staan, vloeien deze op het scherm in elkaar over en ontstaat er een nieuwe kleur. Ter verduidelijking zullen we een eenvoudig voorbeeld geven. We willen een oranje balletje maken. U moet daarvoor twee verschillende balletjes maken. Het eerste balletje moet geheel ingekleurd zijn. Het tweede balletje moet dezelfde afmeting hebben als de eerste, maar moet in plaats van geheel ingekleurd een tralie zijn (zie figuur 4). Zoals u ziet bestaat het tweede balletje om en om uit vertikale rijen bits die aan staan en vertikale rijen bits die uit staan. Door het eerste balletje de kleur rood te geven en het tweede balletje geel te kleuren en vervolgens het gele balletje het rode te laten overlappen, ontstaat een oranje balletje. Deze kleur oranje is anders van teint dan de standaard oranje van de Commodore. Deze methode om kleuren te mengen werkt alleen met single color sprites. Bij multi color sprites zouden de spijlen van de tralie te breed zijn, waardoor de kleuren niet geheel in elkaar overvloeien. In het begin van deze paragraaf hebben we gezegd dat we 136 kleuren kunnen maken. Dit getal is als volgt berekend. Elke van de 16 standaard kleuren kunt u met 15 andere kleuren mengen. Zo zijn er dus 15*16=240 verschillende combinaties te maken. Elke combinatie is echter 2 keer geteld, immers groen met blauw geeft dezelfde mengkleur als blauw met groen. In totaal zijn er dan 240/2=120 mengkleuren te maken. Samen met de 16 standaard kleuren levert dit 236 verschillende kleuren op. Het volgende programma laat een balk met vijf verschillende teinten blauw zien.

```
10 rem gemengde kleuren
20 poke 53280,0:poke 53281,0
30 print chr$(147)
40 for i=832 to 894:poke
   i,255:next
50 for i=896 to 958:poke
   i,170:next
60 for i=0 to 7
70 read a:poke 2040+i,a
80 read a:poke 53287+i,a
90 read a:poke 53248+i*2,a
 100 poke53249+i*2,100:next
110 poke 53264,0:poke53269,
    255
120 data 14,1,196,13,14,196,
    14, 3,172
130 data 13,14,172,13,14,148,
    14,14,124
140 data 13,6,124,13,6,100
```



Figuur 4

Applikatie programma's

Het eerste applikatie programma is een sprite-extractor. Dit programma stelt u in staat om sprites uit programma's te extraheren, zodat u deze in uw eigen programma's kunt gebruiken. Om een bepaalde sprite uit een programma te halen, gaat u als volgt te werk. Eerst laadt u het programma in de computer en runt het. Zodra het programma gestart is, reset u de computer. Vervolgens leest u de sprite-extractor in en runt deze. Boven op het scherm staat een sprite. Onder deze sprite staat het beginadres van de bytewaarden van deze sprite. Door op de F1-toets te drukken, kunt u de sprite die 64 bytes verder in het geheugen staat, laten zien. Met de F3-toets kunt u weer terug gaan naar de vorige sprite. Met deze toetsen zoekt u dan het hele geheugen af, tot u de juiste sprite gevonden heeft. Vervolgens drukt u op F7 om de bytewaarden te zien. Deze bytewaarden kunt u dan verwerken in uw eigen programma. Helaas kunt u op deze manier niet altijd de gewenste sprite vinden, omdat de sprite-extractor een stuk van het geheugen, waar eventueel sprites staan, gebruikt. Om dit euvel te verhelpen zou u het programma moeten herschrijven in machinetaal en deze in het geheugengebied tussen 4096 en 8192 plaatsen. In dit geheugengebied staan namelijk nooit sprites (zie vorige aflevering). Het tweede applikatie programma is een multi color sprite editor. Dit programma is en werkt in grote lijnen hetzelfde als de single color sprite editor van vorige keer. We zullen de werking van het programma voor alle duidelijkheid hier uitleggen. Als u het programma gerund heeft moet u eerst het spritenummer ingeven van de sprite die u wilt ontwerpen. Vervolgens moeten de drie kleuren van de sprite ingegeven worden in de vorm van getallen. Nadat op het scherm een raster is verschenen, kunt u de sprite gaan ontwerpen met behulp van de joystick. U kunt met de joystick de cursor in het raster bewegen. De cursor zelf kan in vijf modes staan. Met de eerste mode kunt u pixels wissen, daarna komen drie modes waarmee u pixels van de sprite aan kunt zetten; voor elke kleur is er een mode. Met de cursor in de laatste, vijfde mode kunt u gewoon met de cursor over alle pixels heen bewegen, zonder dat ze veranderen. U kunt van mode veranderen door op de vuurknop van de joystick te drukken. De sprite die u aan het ontwerpen bent, staat naast het raster op het beeld en verandert mee met de verbeteringen die u aanbrengt aan de sprite. Als laatste zitten er nog wat opties onder de numerieke toetsen. Als u klaar bent met de sprite kunnen, door '1' in te drukken, de bytewaarden van de sprite worden verkregen.

Dit was het voor deze aflevering. We denken dat we nu ongeveer de basis elementen van de sprites allemaal hebben behandeld en hopen dat u er een beetje mee uit de voeten kan. In een latere aflevering over animatie komen we nog uitgebreid terug op de fascinerende wereld van de sprites en zullen dan ook dieper ingaan op de vele toepassingen die de sprites hebben in allerlei games en animaties. De volgende aflevering zal gaan over de moeilijkste van de drie grafische objecten die de Commodore 64 rijk is, namelijk high resolution. Tot de volgende keer.

Hylke Sprangers & Michel de Boer

Listing 1

```
10 rem *******************
15 rem *                 *
20 rem * sprite-extractor *
25 rem *                 *
30 rem *******************
40 rem
44 poke 55,0:poke 56,79:clr
45 ad=0:mc=0
```

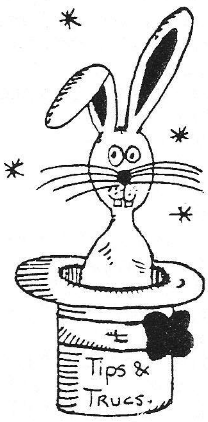
```
50  for i=0 to 33:read a
60  s=s+a:poke 20224+i,a:next
70  if s<>4691 then print"fout
    in data!":end
80  poke 53280,0:poke
    53281,0:poke 646,7
90  print chr$(147)spc(12)
    "sprite-extractor"
110   poke 53248,172:
    poke53249,74
120   poke 53287,15:poke
    53285,11:poke 53286,12
130   poke 53276,mc:sys
    20224,ad
140   poke 2040,11:poke 53269,1
150   print:print
    spc(17)"++++++"
160   for i=1 to 3
170   print spc(17)"+
    +":next
180  print spc(17)"++++++"
190  print "qqqqqq"spc(10)"f1
    volgende sprite"
200   print spc(10)"f3 vorige
    sprite"
210   print spc(10)"f5
    multicolor aan/uit"
220   print spc(10)"f7 byte
    waarden"
230   gosub 500:get a$
240   if (a$<chr$(133)) or
    (a$>chr$(136)) then 230
250   a=asc(a$)-132:poke 198,0
260   on a goto 300,320,340,350
300   ad=ad+64:if ad>65535 then
    ad=0
310 sys20224,ad:goto 230
320   ad=ad-64:if ad<0 then
    ad=65472
330   sys20224,ad:goto 230
340   mc=1-mc:poke
    53276,mc:goto 230
350   poke 53269,0
355   print chr$(147)"byte
    waarden"
360   for i=0 to 62 step 3
370   for j=0 to 2
380   print peek(704+i+j),:next
390   print:next
400   print"druk een toets!"
410   wait 198,1:poke 198,0
420   goto 80
500   poke 781,9:poke 782,15
505   poke 783,peek(783) and
    254
510   sys 65520
520   print"adres:";str$(ad);""
530   return
1000  data 32,253,174,32,158,
    173, 32,247
1010  data 183,132,251,133,
    252, 120,160,0
1020  data 132,1,177,251,153,
    192,2,200
1030  data 192,63,208,246,169,
    55,133,1
1040  data 88,96
```

Listing 2

```
10  rem ******************
20  rem *                *
30  rem *  multi - color  *
40  rem * sprite - editor *
50  rem *                *
60  rem *******************
70  rem
80  rem initialisatie
90  rem
100   poke 53281,254:poke
    53280,254
110   for x=1 to 5:read u1(x)
120   read u2(x):next:dim
    js(16)
130   for x=6 to 15:read
    a:js(x)=a:next
140   for x=0 to 7:poke
    2040+x,248+x
150 next:for x=0 to 130
160   read a:poke
    49152+x,a:next
170   rem
180   rem sprite & kleuren
    kiezen
190   rem
200   print chr$(147)chr$(5);
210   input"spritenummer
    (0-7)";sn
220   if sn<0 or sn>7 then 210
230   xc=1079:qq=32:cu=1:poke
    53276,2^sn
240   for x=2 to
    4:print"kleur"x-2;
250   input u3(x):poke
    53283+x,u3(x):next
260   poke 53287+sn,u3(4)
270 ga=(248+sn)*64:h=int(
    ga/256)
280   l=ga-256*h:poke
    253,l:poke 254,h
290   poke 53248+sn*2,50
300   poke 53249+sn*2,150
310   print tab(40)"joystick in
    port 2"
320   rem
330   rem schermopbouw
340   rem
350   for q=0 to
    2000:next:print chr$(147)
360   for q=1038 to
    1051:pokeq,90
370   poke q+880,90:next
380   for q=1038 to 1918 step
    40
390   poke q,90:poke
    q+13,90:next
400   print"1: sprite af"
410   print"2: beeld leeg"
420   print"3: x,y klein"
430   print"4: x groot"
440   print"5: y groot":poke
    53269,2^sn
450   rem
460   rem joystick &
    toetsenbord uitlezen
470   rem
480   get a$:sys 49152:poke
    xc,u1(cu)
490   poke xc+54272,1:for t=0
    to 70:next
500   if a$="1" then 670
510   if a$="2" then gosub 740
520   if a$="3" then poke
    53271,0:poke 53277,0
530   if a$="4" then poke
    53277,2^sn
540   if a$="5" then poke
    53271,2^sn
550   a=peek(56320):ri=js
    (abs(a-111))
560   if a=111 then 610
570   if ri=0 or peek(xc+ri)=90
    then 480
580   u2(5)=qq:qq=peek
    (xc+ri):u3(5)=qr
590   qr=peek(xc+ri+54272):poke
    xc,u2(cu)
600 poke xc+54272,u3(cu):xc=
    xc+ri:goto 480
610   cu=cu+1:if cu=6 then cu=1
620   goto 480
630   rem
640   rem opties uitvoeren
650   rem
660   poke xc,qq:sys 49152
670   print chr$(147):poke
    53269,0
680   for x=0 to 20:printx":";
690   for y=0 to 2:print
    peek(x*3+y+ga);
700   next:print:next
710   print tab(40)"druk een
      toets in."
720   get a$:if a$="" then 720
730   goto 200
740   a=1079:for x=0 to 20:for
    t=0 to 11
750   poke
    a+t,32:next:a=a+40:next:ret
    urn
760   rem
770   rem data getallen
780   rem
790   data 86,32,160,160,81
800   data 81,102,102,43,32
810   data 41,-39,1,0,39,-41,
    -1, 0,40,-40
820   data 169,0,141,252,
    207,169,4
830   data 133,252,169,55,133,
    251,160
840   data 0,169,0,141,253,
    207,162
850   data 3,177,251,201,81,
    208,9
860   data 32,103,192,32,113,
    192,76
870   data 54,192,201,160,208,
    6,32
880   data 103,192,76,54,192,
    201,102
890   data 208,3,32,113,192,
    202,200
900   data 224,255,208,218,152,
    72,172
910   data 252,207,173,253,207,
    145,253
920   data 104,168,238,252,207,
    192,12
930   data 208,192,24,165,251,
    105,40
940   data 133,251,165,252,105,
    0,133
950   data 252,201,7,208,173,
    165,251
960 data 201,127,208,167,96,
    189,127
970 data 192,13,253,207,141,
    253,207
980 data 96,189,123,192,13,
    253,207
990 data 141,253,207,96,
    2,8,32
1000 data 128,1,4,16,64
```

Deze keer in Tips & Trucs enkele wetenswaardigheden over de datarecorder, waarmee u een optimaler gebruik kunt maken van dit opslagmedium. Verder een tip ter verbetering van de interactie tussen Basic programma's en machinetaal routines; en natuurlijk weer peeks & pokes. Lezers die zelf handige Tips & Trucs hebben worden van harte uitgenodigd deze in te sturen.

Tips & Trucs 64



De tot nu toe behandelde tips & trucs zijn allen afkomstig uit onze eigen truccendoos. Helaas is deze truccendoos geen bodemloze put en daarom roepen we alle lezers op om eigen tips & trucs in te sturen. Stuur alles naar Commodore Info, Postbus 43048, 1009 ZA, Amsterdam onder vermelding van Tips & Trucs 64.

De Datarecorder

Met de datarecorder van de Commodore 64 kunt u gegevens bewaren. De Computer maakt verschil tussen programma's en bestanden. Onder een programma wordt een reeks instrukties in een computertaal (hier Basic) verstaan. Een bestand bevat louter gegevens in de vorm van getallen of strings. We zullen het hebben over het bewaren van programma's op tape. Het schrijven van een programma op tape kunt u doen met het commando SAVE. In figuur 1 staan de verschillende manieren om een programma te saven. Het inladen van een programma van tape in de computer kan worden gedaan met het commando LOAD. Ook van LOAD staan de verschillende manieren om dit te doen uitgewerkt in figuur 1.

Elk randapparaat heeft een eigen device-nummer. Door middel van dit nummer kan de computer de verschillende apparaten uit elkaar houden. Zo heeft de disc-drive device-nummer 8 en de datarecorder heeft nummer 1. Als u twee disc-drives heeft, kunt u voor de tweede drive device-nummer 9 gebruiken. Bij de commando's LOAD en SAVE kan dit nummer meegegeven worden. Als bij LOAD en SAVE geen device-nummer meegegeven wordt, wordt als apparaat de datarecorder genomen. Zo is LOAD "FILE" hetzelfde als LOAD"FILE",1. Zoals u ziet moet het device-nummer achter de naam worden meegegeven.

Verschillende manieren van SAVE en LOAD	
save	save zonder naam naar tape
save"prog"	save 'prog' naar tape
save"prog",1	save 'prog' naar tape
save"prog",8	save 'prog' naar disc
save"prog",1,1	save, niet verplaatsbaar
load	laadt eerste file van tape
load"*",8	laadt eerste file van disc
load"prog",1	laadt 'prog' van tape
load"prog",8	laadt 'prog' van disc
load"$",8	laadt disc directory
load"prog",1,1	laadt, niet verplaatsbaar
load"prog	laadt file van tape die begint met 'prog'
load"prog*",8	laadt file van disc die begint met 'prog'

Figuur 1

Als er een programma moet worden geladen van tape, moet de computer eerst gaan zoeken waar het programma zich bevindt op tape. Als het begin van het programma eenmaal gevonden is, moet de Commodore ook nog kijken of het goede programma wel gevonden is (in het geval dat er een naam bij het LOAD commando is meegegeven). De Commodore 64 doet dit aan de hand van pattern matching. Dit is een techniek om programma en bestandsnamen op tape en op disc te herkennen. Als de afsluitende aanhalingstekens na de programmanaam bij LOAD worden weggelaten, zoekt de computer naar een programma waarvan de naam begint met de opgegeven naam.

```
LOAD"DEEL
```

Het voorbeeld hierboven zal tot gevolg hebben dat de computer het eerste programma laadt van tape dat met DEEL begint. Zo zou het kunnen zijn dat de computer een van de volgende programma's laadt: DEEL1, DEEL2. Dit in tegenstelling tot het commando LOAD"DEEL" dat specifiek naar een programma zoekt met als naam DEEL.

Geheugengebieden

Het laden en saven van willekeurige gebieden in het geheugen kan heel simpel op de Commodore. U wilt bij-

voorbeeld een sprite die ergens in het geheugen staat, of bijvoorbeeld een machinetaal programma saven. Voor u dit kunt doen moet u eerst iets meer weten over de zero-page adressen 43 tot 46. De eerste 2 adressen, 43 en 44, vormen samen het beginadres waar een Basic programma begint. Deze twee adressen worden samen een pointer genoemd. In dit geval wijst de pointer naar het begin van de Basic geheugenruimte. Daarbij is 44 het hi- en 43 het lo-byte. Het beginadres wordt dan als volgt berekent: beginadres = peek(43)+256*peek(44). Normaal wijst deze pointer naar adres 2049, het adres waar standaard de Basic programma's beginnen. De twee adressen 45 en 46 geven het eindadres aan van het Basic programma dat in de computer zit. De lengte van het Basic programma dat in de computer zit, bepaalt dus het adres waar deze pointer naar wijst: eindadres = peek(45)+256*peek(46). Overigens geeft deze pointer ook het adres aan waar de opslag begint van variabelen die in een Basic programma worden gebruikt. Dit, omdat de variabelen worden opgeslagen direct na het programma zelf. Als u gewoon een Basic programma op tape wegschrijft met SAVE wordt het geheugengebied weggesaved dat begrensd wordt door de geheugenadressen waar de twee genoemde pointers naar wijzen.

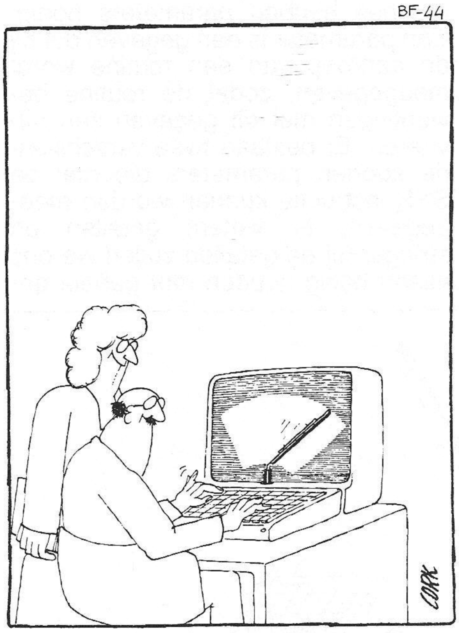
Nu willen we een willekeurig geheugengebied op tape wegschrijven. Wat we dus moeten doen, is de twee pointers naar het begin- en eindadres van het te saven gebied laten wijzen. Als voorbeeld zullen we het scherm kiezen om weg te saven. Het beginadres van het scherm is adres 1024. De hi- en lo-bytes worden nu als volgt berekend: hi-byte = int(beginadres/256), lo-byte = beginadres-256*hi-byte. Bij het beginadres van het scherm is het hi-byte 4 en het lo-byte 0. Deze twee getallen moeten respectievelijk in adres 44 en 43 worden gepoked. Hetzelfde moet daarna ook gebeuren voor het eindadres van het scherm: 2023. Hiervan is het hi-byte int(2023/256)=7 en het lo-byte 2023-256*7=231. Deze twee getallen moeten achtereenvolgens in adressen 46 en 45 worden gezet.

Als deze twee pointers naar de juiste adressen wijzen, kan het geheugengebied worden gesaved. Bij het saven moet echter nog wel een zogenaamd sekundair adres worden meegegeven. Het volgende voorbeeld maakt dit wat duidelijker:

```
SAVE"NAAM",1,1
```

De eerste 1 bij het SAVE commando is het device-nummer. Dit is een 1 omdat we op tape willen saven. De tweede nummer is het sekundaire adres. Als u een Basic programa saved met SAVE"NAAM" is dit adres 0.

BF-44



Dit sekundaire adres heeft gevolgen voor het laden van het programma. Een programma dat met sekundair adres 0 is gesaved, kan op twee manieren worden geladen. Het programma wordt teruggeladen op het adres dat in de adressen 43 en 44 staat, als het sekundaire adres bij het laden 0 is (LOAD"NAAM",1,0); normaal is dit 2049. Het is hierdoor mogelijk om programma's die gesaved zijn met het sekundaire adres 0 in een willekeurig geheugengebied te laden door het adres in 43 en 44 te veranderen. Het programma is verplaatsbaar. Als echter bij het laden een sekundair adres 1 wordt meegegeven (LOAD"NAAM", 1,1) dan wordt het programma ingeladen op het adres, waarvandaan het gesaved was.

De tweede manier om een programma te saven is met het sekundaire adres 1. Dit betekent dat het gesavede geheugengebied altijd in hetzelfde gebied wordt teruggeladen, ongeacht het sekundaire adres bij het laden. Het programma is dus niet verplaatsbaar. Nu we dit allemaal weten kunnen we heel simpel een stuk geheugen saven. Het volgende programma vraagt om een begin en en een eindadres van een geheugengebied en een naam, en schrijft vervolgens het gebied weg op tape.

```
10  rem save geheugengebied
20  lb=peek(43):hb=peek(44)
30  le=peek(45):he=peek(46)
40  input"beginadres";ba
50  hi=int(ba/256):lo=ba-
    256*hi
60  poke 43,lo:poke 44,hi
70  input"einadres";ea
80  hi=int(ea/256):lo=ea-
    256*hi
90  poke 45,lo:poke 46,hi
100   input"naam";na$
110   save na$,1,1
120   poke 43,lb:poke 44,hb
130   poke 45,le:poke 46,he
```

De Cassettebuffer

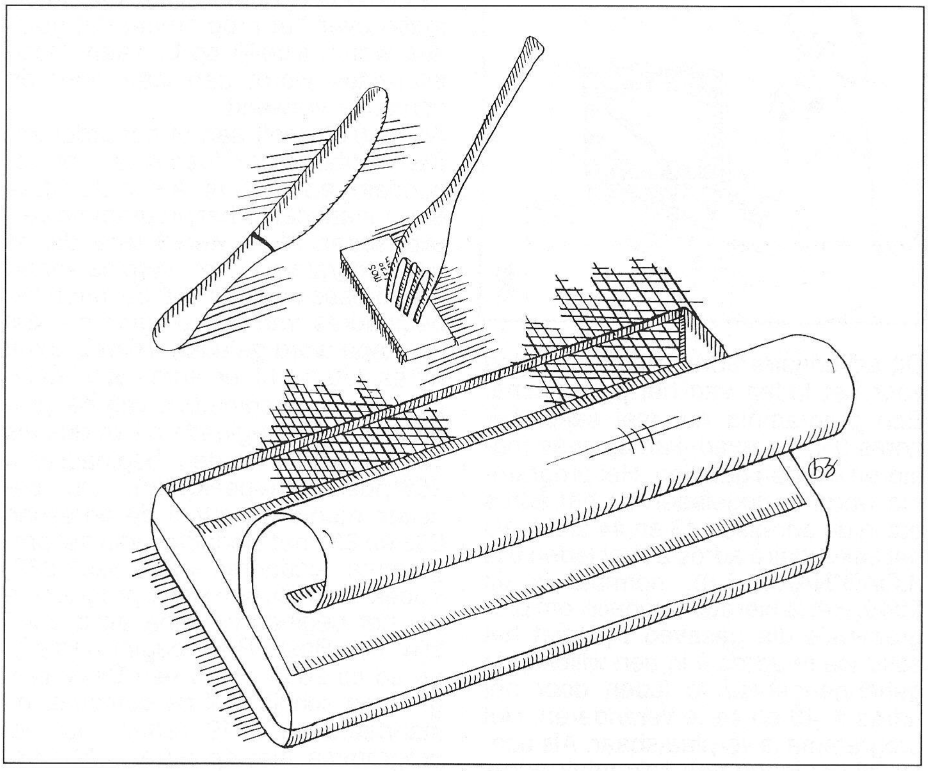
Het geheugengebied van 828 tot 1019 in de computer bevat de cassettebuffer. De cassettebuffer wordt door de computer gebruikt om allerlei informatie over het programma dat geladen wordt, tijdelijk op te slaan. Deze informatie wordt dan later door de computer verwerkt.

Adres 828 geeft aan of het programma verplaatsbaar (gesaved met sekundaire adres 0) is. Als in dit adres een 1 staat,dan is het programma verplaatsbaar. Als er een 3 staat dan is het programma niet verplaatsbaar. De adressen 829 en 830 bevatten het beginadres van het programma dat van tape werd geladen. Hierbij bevat adres 830 het hi- en adres 829 het lobyte van het beginadres van het programma. Het beginadres kan dus als volgt berekent worden: beginadres = 256*peek(830)+peek(829). Op dezelfde manier bevatten de adressen 832 en 831 het eindadres van het programma: eindadres = 256*peek(832) +peek(831). Bij een Basic programma zal het beginadres bijna altijd 2049 zijn. Het Basic RAM begint namelijk op adres 2048. Als u een Basic programma schrijft, zal de computer dit standaard op 2049 zetten. Dat het programma niet op adres 2048 begint, komt door het feit dat het byte voor het beginadres van het Basic programma altijd een nul moet bevatten. Adres 2048 bevat dus een nul. Bij een machinetaalprogramma kan het beginadres verschillend zijn, en kunnen de adressen 829 en 830 goed van pas komen.

De adressen 833 tot 1019 bevatten de naam van het ingeladen programma. De naam wordt in ASCII codes in de adressen gezet. Adres 183 bevat de lengte van de naam. Als de lengte groter dan 16 is, zal de computer bij de SEARCHING en FOUND boodschap slechts de eerste 16 karakters afdrukken. Met het volgende statement kunt u de naam van het programma op het beeld krijgen.

```
FOR X=833 TO 832+PEEK(183):
    PRINT CHR$(PEEK(X));:
    NEXT X
```

Als u geen programma hoeft te laden van tape, kunt u de cassettebuffer mooi gebruiken, om er allerlei gegevens in te zetten. Vaak wordt dit gebied van het geheugen gebruikt om er sprites in te zetten. U kunt in het hele gebied maximaal drie sprites zetten. De drie sprites kunnen op respectievelijk de adressen 832 (13*64), 896 (14*64) en 960 (15*64) worden neergezet. U moet wel uitkijken met het zetten van gegevens in de cassettebuffer, want als er een programma wordt geladen, gaan de sprites voorgoed verloren.



Het volgende programma zoekt het eerst volgende programma op tape en print daarvan alle gegevens in de cassettebuffer op het scherm.

```
10  rem gegevens programma op
    tape
20  open 1,1,0,""
20  ba=peek(829)+256*peek(830)
30  ea=peek(831)+256*peek(832)
40  for x=833 to 832+peek(183)
50  na$=na$+chr$(peek(x)):next
60  if peek(828)=1 then print
    "verplaatsbaar"
70  if peek(828)=3 then print
    "niet verplaatsbaar"
80  print"beginadres:"ba
90  print"eindadres:"ea
100   print"naam:"na$
```

SYS met parameters

Bij het programmeren in BASIC worden vaak kleine machinetaal routines gebruikt, omdat het programma anders te langzaam wordt, of omdat bepaalde problemen gewoon niet in BASIC kunnen worden geprogrammeerd. Veel van zulke machinetaal routines hebben parameters nodig. Een parameter is een gegeven dat bij de aanroep van een routine wordt meegegeven, zodat de routine bewerkingen met dit gegeven kan uitvoeren. Er bestaan twee verschillende soorten parameters die met de SYS instructie kunnen worden meegegeven, te weten: getallen en strings. Bij de getallen zullen we ons alleen bezig houden met gehele getallen. De parameters moeten op de een of andere manier aan het machinetaal programma worden doorgegeven.

Als voorbeeld van een machinetaal routine, die parameters nodig heeft, nemen we een routine die een string op een bepaalde positie op het scherm afdrukt. De schermpositie bestaat hierbij uit een regel- en een kolomnummer. De bovenste regel is regelnummer 0 en de linker kolom is kolomnummer 0. De routine heeft dus een string, een regelnummer en een kolomnummer als parameters. Hoe moeten we deze parameters nu vanuit een BASIC programma aan de machinetaal routine doorgeven? De eenvoudigste manier is om alle parameters op bepaalde adressen in het geheugen te poken en vervolgens met de SYS instructie de routine aanroepen. Deze kan dan de adressen uitlezen om te kijken wat de parameters zijn. Deze methode is echter vrij onhandig om de volgende redenen:

° De programmeur moet onthouden in welke adressen hij welke parameters moet poken.
° Bij getallen groter dan 255 moet het getal gesplitst worden in een lo- en een hi-byte, omdat een geheugenadres slechts getallen tussen 0 en 255 kan bevatten.
° Bij strings moeten alle karakters uit de string een voor een als ASCII-code in het geheugen worden gepoked. Daardoor wordt bij lange strings het programma traag.
° Door de vele pokes, die nodig zijn om de parameters door te geven, wordt het programma trager.

Wij zullen daarom een methode aan de hand doen, die deze problemen niet heeft. Met deze methode kunnen de parameters gewoon met de SYS instructie worden doorgeven. De voorbeeld routine kan dan op de volgende wijze worden aangeroepen: SYS 49152,"commodore info",10,12. De routine zal dan de tekst "commodore info" op positie 10,12 (rij,kolom) afdrukken.

Met behulp van een aantal ROM routines kunnen we vanuit de machinetaal routine de parameters achter de SYS instructie inlezen. We zullen deze routines een voor een bespreken:

- $AEFD: Door deze routine aan te roepen wordt een komma weggelezen.
- $AD9E: Deze routine leest een parameter in.

Als de parameter een string is, dan staat in de adressen $22/$23 het adres waarin het eerste karakter van de string staat. De rest van de string staat in de opvolgende adressen. Aan het eind van de karakters staat een 0 in het geheugen, zodat het einde van de string kan worden bepaald.

Als de parameter een getal is dan staat dit getal in een zogenaamde floating point accumulator. (Over deze accumulator zouden we een heel artikel kunnen schrijven. We zullen daarom niet uitleggen wat dit is). Met behulp van verderop beschreven routines kan de ingelezen parameter worden omgezet in een lo- en een hi-byte, zodat we niks meer te maken hebben met de floating point accumulator.

- $B79E: Deze routine leest een getal-parameter, die tussen 0 en 255 ligt. De gelezen parameter wordt in het X-register gezet.
- De volgende twee routines zetten een parameter die is ingelezen met $AD9E om in een lo- en een hi-byte.
- $B1BF: Deze routine zet een ingelezen parameter, die tussem -32768 en 32767 ligt, om. Het hi-byte wordt in adres $64 gezet en het lo-byte in $65.
- $B7F7: Deze routine zet een ingelezen parameter, die tussen 0 en 65536 ligt, om. Het Y-register en de accumulator bevatten respectievelijk het lo- en het hi-byte.

Met deze ROM routines kunnen we nu de voorbeeld routine, om een string op een bepaalde plaats op het scherm te zetten, programmeren. Als eerste moeten we een komma skippen. Dit doen we met JSR $AEFD. Vervolgens moet er een string parameter gelezen worden. Dit kan met JSR $AD9E. Het adres van de gelezen string staat nu in $22/$23. Dit adres moeten we ergens anders bewaren omdat $22/$23 gebruikt worden door de ROM routines die de volgende parameters lezen. Dan skippen we weer een komma met JSR $AEFD. Nu moet er een getal parameter (het regelnummer) gelezen worden. Omdat dit getal altijd tussen 0 en 255 ligt (er zijn maar 25 regels) kunnen we JSR $B79E gebruiken. Het regelnummer staat nu in het X-register. Daarna skippen we weer een komma en moet er nog een getal parameter (het kolomnummer) gelezen worden. Dit zouden we weer kunnen doen met JSR $B79E, maar om ook te laten zien hoe een getal met de routine op $AD9E gelezen kan worden, zullen we deze routine nu gebruiken. Na JSR $AD9E staat het kolomnummer in een floating point accumulator (FAC). Dit kolomnummer zetten we vervolgens om in een lo- en een hi-byte met JSR $B7F7. Aangezien het grootste kolomnummer 39 is, is het hi-byte altijd 0 en bevat het Y-register dus het kolomnummer. Hieronder volgt de assembly-code van de routine:

Labels Mnemonics Commentaar

```
jsr     $aefd       ;Skip komma.
jsr     $ad9e       ;Lees string in.
lda     $22         ;bewaar string
sta     string      ;pointer.
lda     $23
sta     string+1
jsr     $aefd       ;Skip komma.
jsr     $b79e       ;Lees integer
                     in X.
stx     regel       ;Bewaar
                     regelnr.
jsr     $aefd       ;Skip komma.
jsr     $ad9e       ;Lees numerieke
                     parameter.
jsr     $b7f7       ;Zet FAC om
                     in integer.
sty     kolom       ;Bewaar kol.nr.
ldx     regel
ldy     kolom
clc                 ;Zet cursor op
jsr     $fff0       ;juiste positie.
lda     string
ldy     string+1
jsr     $ab1e       ;Print string.
rts
string  .word 0
regel   .byt  0
kolom   .byt  0
```

De ROM routine op $FFF0 zet de cursur op een regel en een kolom die in het X- en het Y-register staan. Voor de aanroep van de routine moet de carry flag gewist worden (CLC). De routine op $AB1E print een string eindigend op een 0. Het lo- en hi-byte van het adres van deze string moeten respectievelijk in de accumulator en het Y-register staan.
Door deze routine op adres 49152 in het geheugen te zetten, kunt u met SYS 49152,"commodore info",10,12 de tekst "commodore info" op regel 10, kolom 12 zetten. In plaats van constante parameters kunnen ook Basic variabelen achter de SYS-instructie worden gezet. Het volgende programmaatje geeft hiervan een illustratie.

```
10  a$="commodore info"
20  rn=10:kn=12
30  sys 49152,a$,rn,kn
```

Dit programma doet precies hetzelfde als de SYS-instructie met de constante parameters.

Peeks & Pokes

56: Out of memory error

Bij het saven van grote programma's geeft de computer vaak de melding 'Out of memory error' en weigert hij om het programma te saven. Dit kan in de meeste gevallen worden opgelost door simpelweg POKE 56,208 in te tikken. Als u daarna het programma probeert te saven, zal de computer dit zonder protest doen.

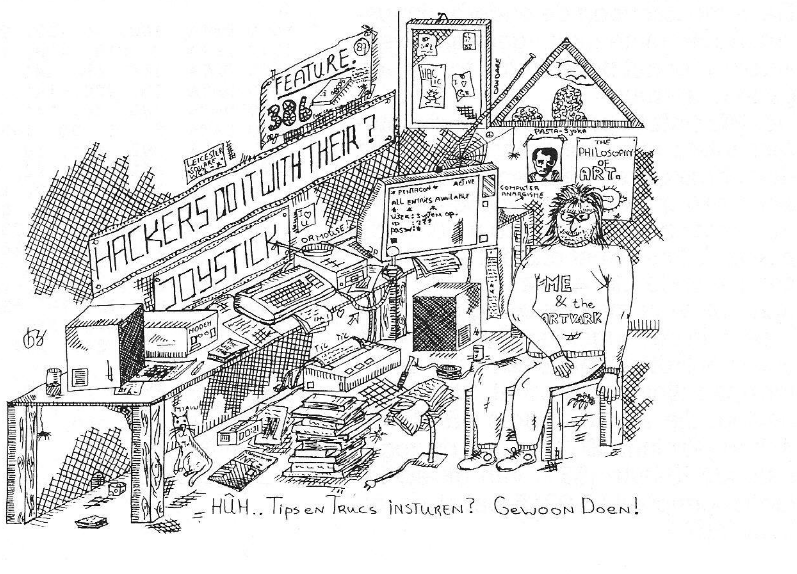
56325: Vertragen

Soms is het handig om de werking van de computer te vertragen. Bijvoorbeeld tijdens het listen van een programma, zodat het programma niet in sneltreinvaart over het scherm vliegt. POKE 56325,0 vertraagt de computer. Door tijdens het listen de SHIFT-toets ingedrukt te houden, gaat het listen nog langzamer.

1: Datarecorder toetsen

In bit 4 van dit adres wordt bijgehouden of er een toets op de datarecorder is ingedrukt. Als de waarde van dit bit 0 is, is er een toets ingedrukt. Als de waarde 1 is, is er geen toets ingedrukt. Door op dit bit te testen kan er een actie worden ondernomen zodra er een toets op de datarecorder is ingedrukt. Vanuit Basic kan dit bijvoorbeeld met de volgende instructie: IF PEEK(1) AND 16 = 0 THEN actie.

Hylke Sprangers en Michel de Boer



Raster interrupts

Wie wel eens een spelletje speelt of een demo bekijkt ziet dan bijvoorbeeld meer dan 8 sprites op het scherm, een border in 5 kleuren, 3 verschillende karaktersets en alsof dit alles nog niet genoeg is hoort men ook nog een alleraardigst muziekje jengelen.Deze effecten zijn in een normaal BASIC-programma niet te realiseren. In machinetaal echter wel, door gebruik te maken van zogenaamde raster-interrupts.

```
10 FORL1=49152TO49170:READQ1:POKEL1,Q1:T1=T1+Q1:NEXTL1:IFT1<>2013THENSTOP
20 PRINT"[illegible]":SYS49152:END
30 :
1000 DATA  120, 169, 13, 141, 20, 3, 169, 192, 141, 21, 3, 88, 96
1010 DATA  238, 32, 208, 76, 49, 234
```

Basic voorbeeld 1

Alvorens te proberen met behulp van raster interrupts zelf een super multicolor (8 kleuren) smooth scrolling te schrijven, zullen we eerst eens kijken hoe de (raster-)interrupt nu eigenlijk werkt.

```
.,C000 78        SEI
.,C001 A9 OD     LDA #$OD
.,C003 8D 14 03  STA $0314
.,C006 A9 CO     LDA #$CO
.,C008 8D 15 03  STA $0315
.,COOB 58        CLI
.,COOC 60        RTS
.,COOD EE 20 DO  INC $D020
.,C010 4C 31 EA  JMP $EA31
```

Object voorbeeld 1

De IRQ

Er zijn verschillende soorten interrupts. Een interrupt is een gestuurde onderbreking van een programma. De computer voert de onderbrekings-/interruptroutine uit en gaat dan weer verder met het programma tot de volgende interrupt.
De IRQ (Interrupt Request) is de interessantste interrupt want hierbij kan de gebruiker zelf bepalen wanneer er een interrupt moet plaatsvinden.
Per seconde vinden er zo'n 60 IRQ's plaats. Dit betekent dus dat elke 1/60-ste seconde de processor het programma waarmee hij bezig is onderbreekt. In geheugenlocaties $0314 en $0315 wordt dan gekeken waar de interruptroutine zich bevindt. Standaard bevindt die zich op adres $EA31. In geheugenplaats $0314 staat de zogenaamde lo-byte ($31) van dit adres. Geheugenplaats $0315 bevat de hi-byte ($EA).

De computer voert nu de interruptroutine vanaf geheugenlocatie $EA31 uit. Deze routine zorgt voor het aftasten van het toetsenbord, de klok (TI$) verhogen e.d.. Wanneer dit gebeurd is keert de computer terug naar het onderbroken programma.
Na al deze theorie is het wellicht tijd voor een "spectaculair" voorbeeld. Voor de mensen die niet over een monitor of assembler beschikken is er het BASIC-programma. Het verdient aanbeveling eventuele cartridges uit te schakelen voordat u voorbeeld 1 of 2 start.

Alles ingetikt? Typ dan SYS49152 ofRUN en de border begint te knipperen dat een stroboscoop er jaloers op zou worden. Ondertussen kunt u gewoon in BASIC verder werken, wat overigens niet aan te raden is als u onvoldoende aspirines in huis heeft.

Wat gebeurt er nu eigenlijk? De instructie SEI zorgt ervoor dat er geen interrupts meer plaatsvinden. Dit is wel handig omdat we nu het adres gaan veranderen waar de computer heenspringt als er een IRQ plaatsvindt. I.p.v. $EA31 gaat de computer nu naar $C00D, omdat we de lo-byte ($0D) van dit adres op $0314 plaatsen en de hi-byte ($C0) op $0315. Als er nu nog wel interrupts zouden plaatsvinden zou dit wel eens kunnen gebeuren als we alleen nog maar de lo-byte hadden veranderd. De processor zou dan naar adres $EA0D springen en "crashen" (niet schrikken, we bedoelen hier niet "crashen" in de betekenis van krakend ineenstorten of te pletter vallen zoals bij vliegtuigen de laatste tijd).
Met CLI wordt het voor de computer weer mogelijk interrupts te genereren. Met RTS tenslotte keert de computer terug naar BASIC.
Elke 1/60-ste seconde springt de computer naar adres $C00D, verhoogt de kleur van de border en springt naar de standaard interrupt

```
10 FORL1=49152TO49372:READQ1:POKEL1,Q1:T1=T1+Q1:NEXTL1:IFT1<>28341THENSTOP
20 FORL2=49408TO49448:READQ2:POKEL2,Q2:T2=T2+Q2:NEXTL2:IFT2<>3533THENSTOP
30 PRINT"[illegible]":SYS49152:END
40 :
1000 DATA  169, 7, 133, 251, 169, 0, 133, 252, 162, 0, 169, 0, 157
1010 DATA  0, 216, 169, 160, 157, 0, 4, 232, 224, 40, 208, 241, 120
1020 DATA  169, 59, 141, 20, 3, 169, 192, 141, 21, 3, 169, 1, 141
1030 DATA  13, 220, 141, 26, 208, 141, 25, 208, 169, 27, 141, 17, 208
1040 DATA  169, 48, 141, 18, 208, 88, 96, 173, 18, 208, 201, 49, 176
1050 DATA  3, 76, 90, 192, 201, 66, 176, 3, 76, 140, 192, 76, 186
1060 DATA  192, 141, 18, 208, 169, 1, 141, 25, 208, 76, 49, 234, 165
1070 DATA  251, 141, 22, 208, 160, 16, 136, 208, 253, 160, 0, 185, 0
1080 DATA  193, 190, 9, 193, 202, 208, 253, 141, 32, 208, 141, 33, 208
1090 DATA  200, 192, 9, 208, 236, 169, 8, 141, 22, 208, 198, 251, 16
1100 DATA  4, 169, 7, 133, 251, 169, 65, 76, 79, 192, 165, 251, 201
1110 DATA  7, 208, 35, 160, 1, 185, 0, 4, 136, 153, 0, 4, 200
1120 DATA  200, 192, 40, 208, 243, 166, 252, 189, 18, 193, 208, 7, 162
1130 DATA  0, 134, 252, 189, 18, 193, 136, 153, 0, 4, 230, 252, 169
1140 DATA  81, 76, 79, 192, 165, 251, 208, 25, 162, 0, 189, 0, 193
1150 DATA  72, 232, 189, 0, 193, 202, 157, 0, 193, 232, 232, 224, 8
1160 DATA  208, 243, 104, 202, 157, 0, 193, 169, 48, 76, 79, 192, 0
1170 :
1180 DATA  13, 3, 14, 6, 4, 2, 10, 7, 0, 8, 1, 8, 8
1190 DATA  8, 8, 8, 8, 8, 131, 143, 141, 141, 143, 132, 143, 146
1200 DATA  133, 160, 137, 142, 134, 143, 160, 160, 160, 160, 160, 160, 160
1210 DATA  160, 160
```

Basic voorbeeld 2

```
.,C000 A9 07    LDA #$07
.,C002 85 FB    STA $FB
.,C004 A9 00    LDA #$00
.,C006 85 FC    STA $FC
.,C008 A2 00    LDX #$00
.,COOA A9 00    LDA #$00
.,COOC 9D 00 D8 STA $D800,X
.,COOF A9 AO    LDA #$AO
.,C011 9D 00 04 STA $0400,X
.,C014 E8       INX
.,C015 EO 28    CPX #$28
.,C017 DO F1    BNE $COOA
.,C019 78       SEI
.,C01A A9 3B    LDA #$3B
.,C01C 8D 14 03 STA $0314
.,C01F A9 CO    LDA #$CO
.,C021 8D 15 03 STA $0315
.,C024 A9 01    LDA #$01
.,C026 8D OD DC STA $DCOD
.,C029 8D 1A DO STA $D01A
.,C02C 8D 19 DO STA $D019
.,C02F A9 1B    LDA #$1B
.,C031 8D 11 DO STA $D011
.,C034 A9 30    LDA #$30
.,C036 8D 12 DO STA $D012
.,C039 58       CLI
.,C03A 60       RTS
.,C03B AD 12 DO LDA $D012
.,C03E C9 31    CMP #$31
.,C040 BO 03    BCS $C045
.,C042 4C 5A CO JMP $C05A
.,C045 C9 42    CMP #$42
.,C047 BO 03    BCS $C04C
.,C049 4C 8C CO JMP $C08C
.,C04C 4C BA CO JMP $COBA
.,C04F 8D 12 DO STA $D012
.,C052 A9 01    LDA #$01
.,C054 8D 19 DO STA $D019
.,C057 4C 31 EA JMP $EA31
.,C05A A5 FB    LDA $FB
.,C05C 8D 16 DO STA $D016
.,C05F AO 10    LDY #$10
.,C061 88       DEY
.,C062 DO FD    BNE $C061
.,C064 AO 00    LDY #$00
.,C066 B9 00 C1 LDA $C100,Y
.,C069 BE 09 C1 LDX $C109,Y
.,C06C CA       DEX
.,C06D DO FD    BNE $C06C
.,C06F 8D 20 DO STA $D020
.,C072 8D 21 DO STA $D021
.,C075 C8       INY
.,C076 CO 09    CPY #$09
.,C078 DO EC    BNE $C066
.,C07A A9 08    LDA #$08
.,C07C 8D 16 DO STA $D016
.,C07F C6 FB    DEC $FB
.,C081 10 04    BPL $C087
.,C083 A9 07    LDA #$07
.,C085 85 FB    STA $FB
.,C087 A9 41    LDA #$41
.,C089 4C 4F CO JMP $C04F
.,C08C A5 FB    LDA $FB
.,C08E C9 07    CMP #$07
.,C090 DO 23    BNE $COB5
.,C092 AO 01    LDY #$01
.,C094 B9 00 04 LDA $0400,Y
.,C097 88       DEY
.,C098 99 00 04 STA $0400,Y
.,C09B C8       INY
.,C09C C8       INY
.,C09D CO 28    CPY #$28
.,C09F DO F3    BNE $C094
.,COA1 A6 FC    LDX $FC
.,COA3 BD 12 C1 LDA $C112,X
.,COA6 DO 07    BNE $COAF
.,COA8 A2 00    LDX #$00
.,COAA 86 FC    STX $FC
.,COAC BD 12 C1 LDA $C112,X
.,COAF 88       DEY
.,COBO 99 00 04 STA $0400,Y
.,COB3 E6 FC    INC $FC
.,COB5 A9 51    LDA #$51
.,COB7 4C 4F CO JMP $C04F
.,COBA A5 FB    LDA $FB
.,COBC DO 19    BNE $COD7
.,COBE A2 00    LDX #$00
.,COCO BD 00 C1 LDA $C100,X
.,COC3 48       PHA
.,COC4 E8       INX
.,COC5 BD 00 C1 LDA $C100,X
.,COC8 CA       DEX
.,COC9 9D 00 C1 STA $C100,X
.,COCC E8       INX
.,COCD E8       INX
.,COCE EO 08    CPX #$08
.,CODO DO F3    BNE $COC5
.,COD2 68       PLA
.,COD3 CA       DEX
.,COD4 9D 00 C1 STA $C100,X
.,COD7 A9 30    LDA #$30
.,COD9 4C 4F CO JMP $C04F
.,CODC 00       BRK

mc100 c12a
.:c100 0d 03 0e 06 04 02 0a 07
.:c108 00 08 01 08 08 08 08 08
.:c110 08 08 83 8f 8d 8d 8f 84
.:c118 8f 92 85 a0 89 8e 86 8f
.:c120 a0 a0 a0 a0 a0 a0 a0 a0
.:c128 a0 00 00 00 00 00 00 00
```

Object voorbeeld 2

routine. De werking van BASIC wordt nu iets vertraagd, omdat de interrupt iets langer is geworden en dus meer tijd kost. Dit is echter voor een normaal mens onmerkbaar. Wie 60 aanslagen per seconde haalt is niet normaal.

Dit voorbeeld geeft duidelijk de snelheid van de IRQ weer. Maar wat heeft u eraan? Al dat geknipper is misschien ideaal voor een acid-house-party; het is niet bepaald een programma waarmee u uw reputatie als programmeertalent bij uw buurman zult bevestigen. De egale verdeling van de kleuren over de border laat wellicht ook wat te wensen over.
U zou een wachtlus in kunnen bouwen, maar het kan handiger. Dit is een taak voor een raster-interrupt.

Voorbeeld 2 maakt gebruik van raster-interrupts. Voorbeeld 2 zullen we nu stapsgewijs ontleden om u de werking van een raster-interrupt duidelijk te maken.

Allereerst, wat moet het programma doen? We willen de border en paper van 9 kleuren voorzien, waarvan er 8 roteren op de eerste regel, waar ook nog eens een tekstje smooth voorbij scrollt. Bij het zien van dit, naar uw zeggen, eenvoudige voorbeeld zal uw buurman spontaan overgaan tot verkoop van zijn PC.

Dewerking

`$C000-$C006:`

In adres $00FB gaan we bijhouden hoever we zijn met scrollen, we beginnen op positie 7. Adres $00FC wordt als tekstwijzer, vrije vertaling van textpointer, gebruikt. Omdat vooraan beginnen in een tekst wel zo mooi is zetten we deze teller op nul.

`$C008-$C017:`

Om aan de scrolling een speciaal effect te geven gebruiken we letters in reverse video. Zwarte letters in reverse video krijgen de kleur van de paper, in dit geval lijkt het dus alsof de letters 8 kleuren hebben die bovendien nog roteren ook. Omdat we eventuele toeschouwers, bijvoorbeeld uw buurman, deze illusie niet willen ontnemen, zorgen we ervoor dat de 8 kleuren van de paper vanaf het begin niet zichtbaar zijn. Dit doen we door de eerste 40 bytes (precies 1 regel) vanaf $D800 (color memory) te vullen met $00 (zwart). De eerste 40 bytes vanaf $0400 (screen memory) vullen we met $A0 (spatie in reverse video). Deze 40 spaties worden dan later door de scrolltekst vervangen.

`$C019-$C03A:`

We verhinderen interrupts door SEI en verleggen de interruptvector naar $C03B op de inmiddels bekend veronderstelde manier.
Voor het gebruik van raster-interrupts moet er echter nog wat meer geregeld worden. Eerst vullen we adres $DC0D met $01 zodat bit 0 van dit re-

```
                    ;   super multi-color smooth
   scrolling
                    ;         voor commodore info
                    ;       (c) 1989   anton loeffen

   ;--------------------------------------
                    ;
                               *= $c000
                    ;
                    irqvctrl = $0314
                    irqvctrh = $0315
                    scrnmem  = $0400
                    colrmem  = $d800
                    msbrstr  = $d011
                    rwraster = $d012
                    scroll   = $d016
                    status   = $d019
                    enable   = $d01a
                    border   = $d020
                    paper    = $d021
                    ciairq   = $dc0d
                    contirq  = $ea31
                    ;
                    scrollpos = $fb
                    txtpointr = $fc
                    ;
c000 a9 07         start     lda #$07
c002 85 fb                   sta scrollpos
c004 a9 00                   lda #$00
c006 85 fc                   sta txtpointr
                             ;
c008 a2 00                   ldx #$00
c00a a9 00         start1    lda #$00
c00c 9d 00 d8                sta colrmem,x
c00f a9 a0                   lda #$a0
c011 9d 00 04                sta scrnmem,x
c014 e8                      inx
c015 e0 28                   cpx #$28
c017 d0 f1                   bne start1
                             ;
c019 78                      sei
c01a a9 3b                   lda #<irq
c01c 8d 14 03                sta irqvctrl
c01f a9 c0                   lda #>irq
c021 8d 15 03                sta irqvctrh
c024 a9 01                   lda #$01
c026 8d 0d dc                sta ciairq
c029 8d 1a d0                sta enable
c02c 8d 19 d0                sta status
c02f a9 1b                   lda #$1b
c031 8d 11 d0                sta msbrstr
c034 a9 30                   lda #$30
c036 8d 12 d0                sta rwraster
c039 58                      cli
c03a 60                      rts
                             ;
c03b ad 12 d0      irq       lda rwraster
c03e c9 31                   cmp #$31
c040 b0 03                   bcs irq2
c042 4c 5a c0                jmp doscroll
                             ;
c045 c9 42         irq2      cmp #$42
c047 b0 03                   bcs irq3
c049 4c 8c c0                jmp domove
                             ;
c04c 4c ba c0      irq3      jmp docolor
                             ;
c04f 8d 12 d0      irqend    sta rwraster
c052 a9 01                   lda #$01
c054 8d 19 d0                sta status
c057 4c 31 ea                jmp contirq
                             ;
                             ;
c05a a5 fb         doscroll  lda scrollpos
c05c 8d 16 d0                sta scroll
                             ;
c05f a0 10                   ldy #$10
c061 88            doscrol1  dey
c062 d0 fd                   bne doscrol1
                             ;
c064 a0 00                   ldy #$00
c066 b9 00 c1      doscrol2  lda color,y
c069 be 09 c1                ldx wait,y
c06c ca            doscrol3  dex
c06d d0 fd                   bne doscrol3
                             ;
c06f 8d 20 d0                sta border
c072 8d 21 d0                sta paper
                             ;
c075 c8                      iny
c076 c0 09                   cpy #$09
c078 d0 ec                   bne doscrol2
                             ;
c07a a9 08                   lda #$08
c07c 8d 16 d0                sta scroll
c07f c6 fb                   dec scrollpos
c081 10 04                   bpl doscrol4
c083 a9 07                   lda #$07
c085 85 fb                   sta scrollpos
                             ;
c087 a9 41         doscrol4  lda #$41
c089 4c 4f c0                jmp irqend
                             ;
c08c a5 fb         domove    lda scrollpos
c08e c9 07                   cmp #$07
c090 d0 23                   bne domove3
                             ;
c092 a0 01                   ldy #$01
c094 b9 00 04      domove1   lda scrnmem,y
c097 88                      dey
c098 99 00 04                sta scrnmem,y
c09b c8                      iny
c09c c8                      iny
c09d c0 28                   cpy #$28
c09f d0 f3                   bne domove1
                             ;
c0a1 a6 fc                   ldx txtpointr
c0a3 bd 12 c1                lda text,x
c0a6 d0 07                   bne domove2
                             ;
c0a8 a2 00                   ldx #$00
c0aa 86 fc                   stx txtpointr
c0ac bd 12 c1                lda text,x
c0af 88            domove2   dey
c0b0 99 00 04                sta scrnmem,y
c0b3 e6 fc                   inc txtpointr
                             ;
c0b5 a9 51         domove3   lda #$51
c0b7 4c 4f c0                jmp irqend
                             ;
c0ba a5 fb         docolor   lda scrollpos
c0bc d0 19                   bne docolor3
                             ;
c0be a2 00                   ldx #$00
c0c0 bd 00 c1                lda color,x
c0c3 48                      pha
c0c4 e8                      inx
c0c5 bd 00 c1      docolor2  lda color,x
c0c8 ca                      dex
c0c9 9d 00 c1                sta color,x
c0cc e8                      inx
c0cd e8                      inx
c0ce e0 08                   cpx #$08
c0d0 d0 f3                   bne docolor2
                             ;
c0d2 68                      pla
c0d3 ca                      dex
c0d4 9d 00 c1                sta color,x
                             ;
c0d7 a9 30         docolor3  lda #$30
c0d9 4c 4f c0                jmp irqend
                             ;
 ;----------------- data ---------------
                             ;
                             *= $c100
                             ;
c100 0d 03 0e 06 04 02 0a 07 00
                   color     .byte 13,3,14,6,4,2,
    10,7,0
                             ;
c109 08 01 08 08 08 08 08 08
                   wait      .byte 8,1,8,8,8,8, 8,8
c111 08                      .byte 8
                             ;
c112 83 8f 8d 8d 8f 84 8f 92 85 a0 89 8e 86 8f a0
    a0 a0 a0 a0 a0 a0 a0
                   text      .text
   "COMMODORE INFO      "
c129 00                      .byte 0
```

gister gezet is en we geen last meer hebben van een timer A interrupt. Om gebruik te kunnen maken van raster-interrupts moet bit 0 van $D01A gezet worden, wat we dan ook doen. Bit 0 van register $D019 wordt gezet als er een raster-interrupt plaatsvindt. Dit register moet dus telkens "schoon" worden gemaakt voordat er weer een raster-interrupt kan plaatsvinden. Bit 0 van dit register wordt "schoongemaakt" door het te vullen met 1. Dit klinkt misschien raar en dat is het ook, als u wilt weten hoe het precies werkt moet u de technische specificaties van I/O chips maar eens nalezen (rare jongens die I/O chips). Voor meer info over de genoemde en nog te noemen registers wordt u verwezen naar de Programmers Reference Guide.

Het scherm is opgebouwd uit 313 rasterlijnen (0-312). De rasterlijnen 51 t/m 251 omvatten de paper. Elke rasterlijn is weer opgebouwd uit 211 beeldpunten (0-210). We kunnen nu aangeven waar er een raster-interrupt moet plaatsvinden door het regelnummer van de desbetreffende rasterlijn in adres $D012 op te slaan. Wanneer u echter Register $D012 leest zult u niet de waarde vinden die u heeft ingevoerd, maar de rasterlijn waar de computer was met tekenen op het moment dat u het register las. Toch onthoudt de computer de door u ingevoerde rasterlijn en zal op tijd een raster-IRQ genereren.

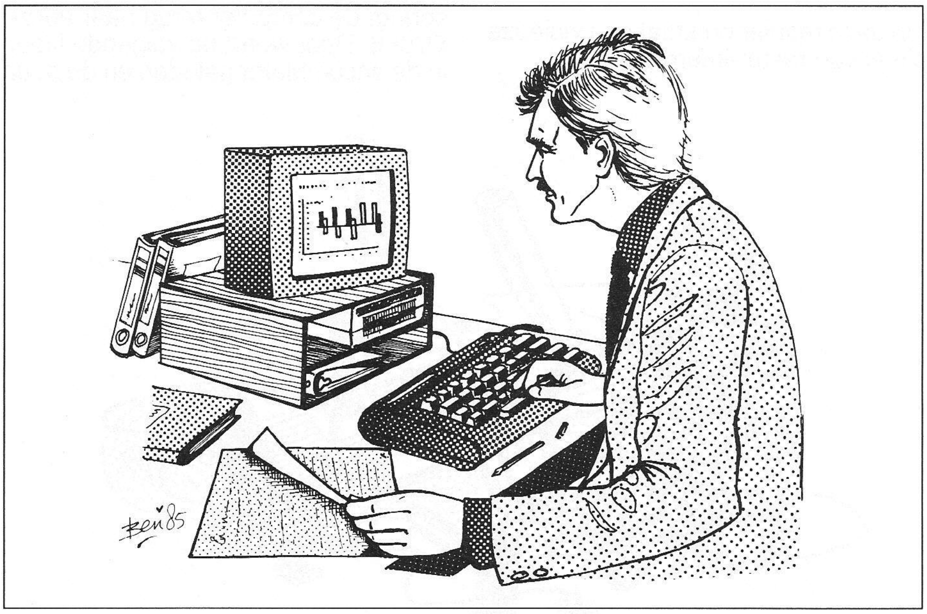
Het scherm bestaat uit 313 lijnen en in een byte kunnen er maar 256 (0-255) worden aangegeven, waar blijven die andere 57 lijnen zult u zeggen. Als de computer bij lijn 256 is aangeland, wordt bit 7 van adres $D011 gezet; is rasterlijn 312 getekend dan wordt bit 7 van dit register weer "gecleared". Wij willen niet dat de eerste raster-IRQ plaatsvindt in het gebied 256-312 dus wordt bit 7 niet gezet. De overige bits hebben een andere functie. Toch maar een Reference Guide kopen?

Wanneer de computer met het tekenen van het scherm is aangeland op het regelnummer in register $D012 wordt er gesprongen naar de routine aangegeven in de IRQ vector ($0314-$0315), in dit geval $C03B en wordt bit 0 van register $D019 gezet. We willen dat de eerste raster-interrupt plaatsvindt als de videocontroller, deze zorgt voor het tekenen vam het scherm, met tekenen is aangeland op regel 48 ($30).

Even recapituleren: we hebben nu de IRQ vector verlegd ($0314-$0315), timer A interrupts uitgeschakeld ($DC0D), we hebben aangegeven dat we met raster-interrupts willen werken ($D01A), het testregister is schoongemaakt ($D019), bit 7 de MSB (Most Significant Bit) van de rasterpositie is "gecleared" ($1B in $D011) en tenslotte hebben we aangegeven waar de eerste raster interrupt moet plaatsvinden ($30 in $D012).

Na CLI zijn er weer interrupts mogelijk en met RTS wordt er teruggekeerd naar BASIC.

`$C03B-$C04C:`



Er vindt een interrupt plaats en de computer springt naar adres $C03B. De accumulator wordt gevuld met de inhoud van $D012. In de accumulator bevindt zich nu de rasterlijn waar de computer was toen er een interrupt plaatsvondt. We hadden ingesteld dat de interrupt moest plaatsvinden op rasterlijn 48 ($30). De inhoud van de accumulator wordt vergeleken met 49 ($31). Omdat 48 nog steeds kleiner is dan 49 wordt er gesprongen naar adres $C05A. Wat deze routine doet bekijken we later, waar het nu om gaat is dat aan het eind van de routine opgegeven moet worden waar de volgende raster-interrupt moet plaatsvinden ($D012) en dat het testregister ($D019) schoongemaakt moet worden.
Terwijl onze interrupt-routine wordt uitgevoerd, gaat de videocontroller gewoon door met het tekenen van het scherm. Aan het eind van de interrupt is de videocontroller met tekenen al minstens 14 rasterlijnen verder. De volgende raster interrupt moeten we dus ook minstens 14 rasterlijnen voorbij rasterlijn 48 laten plaatsvinden. Doen we dit niet dan zal de interrupt worden overgeslagen omdat de vorige interrupt nog niet afgewerkt is. Daarom wordt op $C087 de accumulator geladen met 48 ($30) + 17 = 65 ($41) en wordt er gesprongen naar $C04F.

`$C04F-$C054:`

De inhoud van de accumulator wordt opgeslagen in $D012 en het testregister wordt schoongemaakt. Dit betekent dus dat de volgende raster-IRQ zal plaatsvinden op rasterlijn 65 ($41). De computer is bij deze lijn al klaar met de vorige interrupt en kan dus een nieuwe generen.
Wanneer dit gebeurt, springt de computer weer naar $C03B, gaat weer vergelijken en springt nu naar $C08C omdat, let op, 65 groter is dan 49 maar kleiner dan 66 ($42).

Wat de routine op $C08C doet bekijken we ook later. Aan het eind van deze routine, op $C0B5 wordt de accumulator gevuld met 81 ($51) en wordt weer gesprongen naar $C04F. De volgende raster-interrupt zal dus plaatsvinden op lijn 81.
Wanneer rasterlijn 81 wordt getekend zal er worden gesprongen naar de routine op $C0BA want 81 is zowel groter dan 49 als groter dan 66!!! Ook deze routine wordt wat verderop behandeld. Op $C0D7 wordt de accumulator gevuld met 48 ($30). De volgende raster-IRQ zal nu weer plaats-

vinden op lijn 48 en er wordt weer gesprongen naar $C05A.

Dus, even voor de duidelijkheid:

```
0)  Eerste raster-IRQ op
    rasterlijn 48 ($30, dus
    routine 1);
1)  $C05A-$C089 volgende
    raster-IRQ op lijn 65
    ($42, dus
```

routine 2);

```
2)  $C08C-$C0B7 volgende
    raster-IRQ op lijn 81
    ($51, dus
```

routine 3);

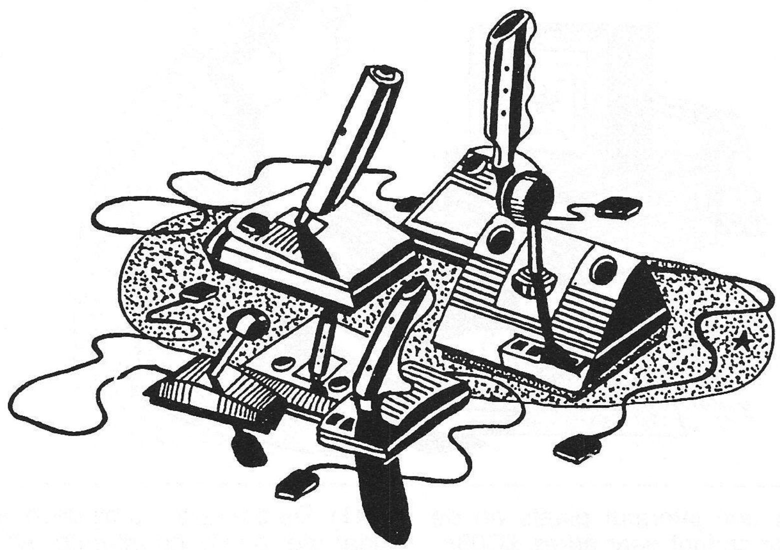
```
3)  $C0BA-$C0D9 volgende
    raster-IRQ op lijn 48
    ($30, dus
```

routine 1).

Op deze manier ontstaat een vicieuze cirkel van raster-interrupts.

Daarom wachten we even m.b.v. de wachtlus op de adressen $C05F tot en met $C062. Dan wordt het Y-register weer op nul gezet. In de accumulator wordt nu de kleur geladen die we in de border en paper willen plaatsen. In het X-register wordt de duur van de wachtlus geladen. Zolang X ongelijk is aan nul springt de computer naar adres $C06C. Op het moment dat X gelijk is aan nul begint de video-controller te tekenen bij beeldpunt 0 op rasterlijn 51. Dit is de eerste rasterlijn binnen de paper. De computer begint nu dus precies met het tekenen van een nieuwe rasterlijn. Op hetzelfde moment veranderen we de kleur van de van de border en paper.

Het Y-register wordt met 1 verhoogd en zolang dit register ongelijk is aan 9 springt de computer terug naar adres $C066. Daar wordt de volgende kleur in de accumulator geladen en de duur



Nu bespreken we (eindelijk) de afzonderlijke routines

`$C05A-$C089:`

De scrollpositie, die werd bijgehouden op zero-page adres $00FB, wordt opgeslagen in adres $D016 . Bit 0 t/m 2 van dit adres zorgen voor het scrollen.

Nu willen we de acht kleuren onder elkaar in de border en paper plaatsen. De videocontroller is ondertussen al verder met het tekenen van het scherm. Zouden we nu de kleur van de border en paper veranderen dan zou het geen mooie recht lijn zijn omdat de processor nog niet alle beeldpunten van de rasterlijn heeft gehad.

van de volgende wachtlus in het X-register. Dit gaat zo door tot negen keer de kleur van border en paper is veranderd. We hebben nu 8 mooie rechte lijntjes in border en paper met elk een verschillende kleur. De belangrijkste voorwaarde voor rechte lijnen is dat de duur van de wachtlus correct is ingesteld. Dit is vooral een kwestie van experimenteren met het invullen van verschillende waarden voor de wachtlus.

De 9 kleuren bevinden zich op adres $C100 t/m C108.
De duur van de wachtlussen bevindt zich op adressen $C109 t/m $C111.
De tekst voor de scroll start op $C112. Zodra er een nul gelezen wordt begint de computer weer vooraan in de tekst.

Is de laatste kleur voor de paper en border, in dit geval zwart, ingesteld dan wordt adres $D016 op 8 gesteld. Het scherm is nu weer normaal 40 koloms. Zouden we dit niet doen dan zou het hele scherm meeschuiven. Hierna wordt de scrollpositie in $00FB met 1 verminderd. Is de waarde voor de scrollpositie niet meer positief, dus kleiner dan 0, dan wordt de scrollpositie weer op 7 gesteld.
De accumulator wordt nu geladen met het rasterlijnnummer voor de volgende raster-interrupt. Er wordt gesprongen naar $C04F waar het lijnnummer ($41) wordt opgeslagen en het testregister schoongemaakt.

`$C08C-$C0B7:`

Wanneer de scrollpositie ($00FB) gelijk is aan 7 worden alle karakters op de eerste regel 1 plaats opgeschoven en wordt er een nieuw karakter uit de scrolltekst gelezen en op het scherm toegevoegd. De tekstwijzer ($00FC) wordt met 1 verhoogd. Zolang de scrollpositie ongelijk is aan 7 wordt meteen doorgesprongen naar $C0B5 en wordt de accumulator gevuld met het lijnnummer voor de volgende raster-IRQ ($51), dan volgt de weer de sprong naar $C04F.

`$C0BA-$C0D9:`

Deze routine zorgt voor het roteren van de eerste 8 kleuren. Dit gebeurt alleen als de scrollpositie gelijk is aan nul.
Is dit niet het geval dan wordt gesprongen naar $C0D7. Het volgende rasterlijnnummer ($30) wordt in de accumulator geladen. Nu volgt de inmiddels bekende sprong naar $C04F met het overbekende vervolg.

Het resultaat

Klaar met tikken? SAVE en dan RUN of SYS49152. Starten maar die super multi-color smooth scrolling. Snel even de buurman halen....

Tot slot

Wanneer u alles een beetje heeft kunnen volgen kunt u nu zelf aan de slag met raster-interrupts. Deze veelzijdige interrupt voegt ongetwijfeld een nieuwe dimensie toe aan uw programma's.

Anton Loeffen

In een vorig nummer van Commodore Info werd reeds melding gemaakt van de applicatie GeoChart. Een pakket, dat in samenwerking met andere GEOS applicaties behoort tot een categorie software van enige allure. In dit artikel gaat Bert Venema in op het importeren van gegevens ter vervaardiging van een grafiek.

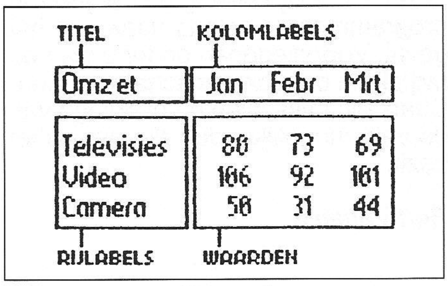
GeoChart (2)

Data krijgen een professioneel uiterlijk

Wanneer u een grafiek wilt maken in GeoChart, dient u eerst de data, de waarden en labels, te vervaardigen. Dit kunt u doen met behulp van één van de andere GEOS applicaties, zoals GeoCalc, GeoFile, GeoWrite of met behulp van de deskaccesoire Note Pad. Als u eenmaal de basis grafiek-data heeft vervaardigd copieert u deze in een zgn. text scrap (tekstknipsel), om het daarna in een GeoChart document in te voegen. Nadat u de grafiek-data in een GeoChart document heeft gekopieerd, bestaat de mogelijkheid om hiervan een grafiek te laten tekenen, of dat u er nog enkele gegevens van te wijzigen, danwel de verschijningsvorm aan te passen.

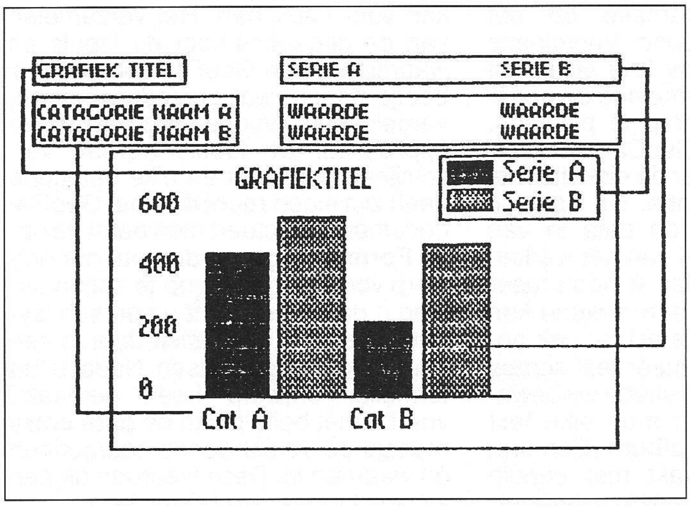
Hoe data zijn opgebouwd

Eerst een korte verhandeling welke gegevens in een **text scrap** moeten zijn vastgelegd, alvorens deze tot een grafiek kunnen worden omgevormd. Elke GEOS-applicatie welke het gebruik van een **text scrap** ondersteunt, is geschikt om grafiekdata te leveren. Het meest voor de hand liggend zijn waarschijnlijk numerieke gegevens, die afkomstig zijn uit een GeoCalc document. In GeoChart zijn de gegevens georganiseerd in series. Elke serie bevat een set van categorieën, met elk hun eigen corresponderende waarden. Het document dat u gaat vervaardigen kan de volgende gegevens bevatten, een grafiektitel, labels voor kolommen, labels voor rijen en waarden. In fig. 1 is afgebeeld hoe de data moeten zijn opgebouwd.



Figuur 1

Elk snijpunt van de label-waarden wordt een **data point** genoemd. In het voorbeeld is het **datapoint** van de televisie's in januari het getal **80**. De maanden **jan**, **febr** en **mrt** zijn labels voor kolommen. De verschillende produkten, zoals televisies, video's en camera's zijn de labels voor de rijen. Aan de linker bovenzijde is de naam van het grafiek in een niet numerieke waarde ingevoerd.



Figuur 2

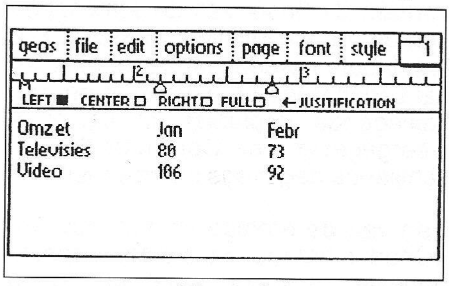
Interpretatie van gegevens door GeoChart

Afhankelijk van de wijze waarop u data heeft geselecteerd, zullen de rij- en kolom-labels aan weerszijden van de grafiek of in een legenda worden weergegeven. In de meeste grafieken worden de rij-labels standaard gelezen als categorie namen en aan de zijkant van de grafiek weergegeven. In de zgn. taart- en unibar grafieken zullen deze gegevens in een legenda worden afgebeeld. De categorieën vormen de basis van de afmetingen van de respectievelijke waarden. Kolom-labels worden standaard gelezen als serienamen en worden in de grafieklegenda geplaatst. In fig. 2 is weergegeven hoe GeoChart de verschillende gegevens interpreteert.

Een van de prettige kanten van GeoChart is het feit, dat als een bepaald gegeven gewijzigd moet worden, u niet een gehele nieuwe dataserie hoeft op te zetten. U kunt gewoon binnen de applicatiewijzigingen aanbrengen. De maximumgrootte van een text scrap is 50 kolommen bij 25 rijen. Indien uw grafiek data verschillende labels bevatten (zoals een titel, catagorienamen, en serienamen), is de maximumwaarde 51 kolommen en 26 rijen.

Vervaardigen van data met GeoCalc

Elke versie van GeoCalc, ook GeoCalc 128 is geschikt om data voor GeoChart te vervaardigen. U voert de gegevens zoals u gewend bent in GeoCalc. Wanneer u grafiek data in GeoCalc invoert, dienen deze elk in een afzonderlijke cel te worden opgenomen. De richting van de gegevens mag echter verschillen. U dient er echter wel voor te zorgen dat de waarden in de zgn. **general** format zijn ingevoerd. Als u een cel leeg wilt laten, moet u de waarde 0 invoeren. U kunt nooit een blanco cel opnemen in de text-scrap. Als u gereed bent met het invoeren van data, kunt u het gebied markeren door deze te **higlighten**. Dit doet u door het begin (linkerbovenzijde) aan te klikken en de muisknop vast te houden terwijl u naar de rechter benedenzijde beweegt. Het gemarkeerde gebied zal in een negatieve weergave op het scherm worden getoond. Vervolgens kiest u de optie **copy text scrap** uit het **options** menu, teneinde de gegevens in een text scrap te plaatsen. Verlaat vervolgens GeoCalc en kopieer de text scrap naar de diskette met de GeoChart-applicatie. Nu opent u GeoChart en leest de data in van GeoCalc. Het nadeel van het werken met text scraps is dat er nooit meer dan één tegelijk op een diskette kan zijn opgeslagen. Er is echter een oplossing, indien men meer text scraps tegelijk op een werkdiskette wil bewaren. Hiertoe kopieert men elke text scrap naar een **text album**. Een **text album** wordt gemaakt met behulp van de desk accesoire **text manager**.



Figuur 3

Aanmaken van data met GeoWrite.

Net als bij GeoCalc is elke versie van GeoWrite geschikt om data aan te maken. Op een nagenoeg gelijke wijze worden de gegevens voor het grafiek in GeoWrite ingevoerd, echter werken we nu niet met cellen. De gegevens, zoals data items, dienen gescheiden te worden door hetzij een **tabstop**, hetzij door komma's. Het einde van elke rij wordt met een RETURN afgesloten, zie fig. 3.

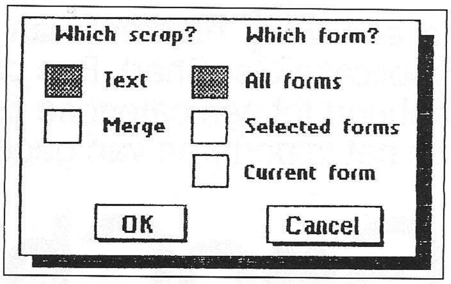
Men dient echter geen spaties tussen elke komma en data-item te plaatsen. Nadat de gegevens zijn ingevoerd, dient het blok dat in een grafiek omgezet moet worden "gehighlighted" te worden, m.a.w. in negatief beeld. Vervolgens wordt de optie **copy** uit het **edit**-menu geselecteerd en worden de gegevens in een text scrap geplaatst. Voor de verdere afhandeling dient de procedure gevolgd te worden, zoals dit bij de methode met GeoCalc al reeds is beschreven.

Geofile data, net iets anders

Net als bij de vorige twee besproken methodes is ook hier elke versie van GeoFile geschikt om data aan te maken voor GeoChart. Het verzamelen van de gegevens voor de labels en waarden in een GeoFile document is beetje anders wat procedure betreft, vergeleken bij vorige twee besproken applicaties. De labels worden verschillend geplaatst en elke categorie heeft zijn eigen **record**. In het GeoFile document selecteert men eerst de optie **Form Design modus** om het ontwerp voor het grafiek op te zetten. Indien u dat wenst, kunt u een sub layout kiezen om de grafiek-data in een specifiek veld te plaatsen. Nadat u het formulier ontwerp heeft gemaakt, voert u met behulp van de **data entry modus** de verschillende categorieën en waarden in. Deze waarden dienen als volgt ingevoerd te worden:

Grafiek Titel	:	**categorie naam A**
Serie naam A	:	**waarde**
Serie naam B	:	**waarde**

en dit doet men precies zo voor de volgende records. Indien u een gegeven blank wilt laten, dient u in het betreffende veld een 0 in te voeren. Nadat de gegevens zijn ingevoerd kunt u een text scrap aanmaken. U dient elk record aan te klikken, welke u in het grafiek opgenomen wilt hebben. Nadat u de noodzakelijke records heeft aangeklikt kiest u de optie **build scrap** uit het **file** submenu. Er verschijnt een dialoogvenster (fig. 4). U klikt op de optie **text** en de optie **all forms**. Tot slot klikt u op het **OK**-icoon waarna de text scrap wordt samengesteld.



Figuur 4

De NotePad methode

Als u GeoChart aanschaft, wordt deze geleverd met de nieuwste versie van de applicatie NotePad. Hierdoor beschikt u over de mogelijkheid om pagina's te verwijderen of toe te voegen, naar andere pagina's te verplaatsen. De eerdere versies van NotePad kennen deze mogelijkheden niet, alsmede de mogelijkheid om text scraps te vervaardigen is eerdere versie's vreemd. De Note Pad applicatie is met name zeer bruikbaar omdat men tijdens het werken met GeoChart nieuwe data aan kan maken. Tevens beschikt Note Pad over de mogelijkheid om maar liefst 127 pagina's aan data op te kunnen slaan. De data wordt op dezelfde wijze ingevoerd als dat dient te gebeuren bij GeoWrite. Om die reden laat ik de verdere werking achterwege. Ten slotte dient een grafiek gemaakt te worden, die altijd uit een text scrap afkomstig moet zijn. Met de voorgaande verhandeling is wederom aangetoond dat de gehele GEOS programmaserie een slim en doordacht concept is. Alle applicaties sluiten op elkaar aan en kunnen onderling gegevens uitwisselen. Men kan in dit verband wel spreken van een zekere integratie van programmatuur. Het is vooral plezierig dat men de applicaties los van elkaar kan aanschaffen en daardoor niet met overtollige programmatuur komt te zitten. Dit is bij volledig geïntegreerde programmatuur helaas vaak wel het geval. Voor diegenen onder u die nog twijfelden over de aanschaf van GeoChart zijn misschien nu overtuigd van de vele mogelijkheden die dit pakket bezit.

Bert Venema.

Basic miniatuurtjes

Een rubriek van Alex van Maarschalkerwaart

Gezocht: Miniatuurtjes
Commodore 64,128,16,4 en Amiga

Stuur al je kleine leuke geinige programmatjes op naar de redactie van Commodore Info, Postbus 43048, 1009 ZA Amsterdam.

Hier volgen een aantal Basic miniatuurtjes gemaakt door Baarda, Van Londen en De Goede.

Uitrollende dobbelstenen

```
 5 rem uitrollende
10 rem dobbelsteen
20 rem baarda en van londen
30 x=2
40 for j=1 to x:next j
50 x=x*1.2+50
60 r=int(6*rnd(0))+1
70 print r; "{SPACE}";
80 if x<150 then goto 40
90 end
```

Old MacDonald 64

```
  10 rem old mac donald 64
  20 rem baarda en de goede
  30 for n=1 to 8
  40 read th(n),tl(n)
  50 next n
  60 poke 54296,15
  70 poke 54277,9
  80 poke 54278,0
  90 read n
 100 if n=0 then goto 180
 110 poke 54276,33
 120 poke 54273,th(n)
 130 poke 54272,tl(n)
 140 for pauze=1 to 200:next pauze
 150 poke 54276,0
 160 for pauze=1 to 25:next pauze
 170 goto 90
 180 for x=54272 to54296:pokex,0:next x
 190 end
 500 data 8,147,9,159,10,205,11,114
 510 data 12,216,14,107,16,47,17,37
1000 data 5,5,5,5,3,3,2,2,7
1010 data 7,6,6,5,5,5,2,5,5
1020 data 5,5,3,3,2,2,7,7
1030 data 6,6,5,5
1040 data 0,0
```

Namen omkeren

```
 10 rem namen omkeren
 20 rem baarda en van londen
 30 print "namen{SPACE}omkeren"
 40 print
 50 print "typ{SPACE}twee{SPACE}namen{
    SPACE}door{SPACE}een{SPACE}komma{S
    PACE}gescheiden"
 60 print "bijv.{SPACE}arleen,oscar"
 70 input a$,b$
 80 print "jouw{SPACE}namen{SPACE}zijn
    {SPACE}..."
 90 print a$;"{SPACE}en{SPACE}";b$
100 print
110 r$=a$:print "reserve{SPACE}krijgt{
    SPACE}eerste{SPACE}naam{2xSPACE}={
    SPACE}";r$
120 a$=b$:print "eerste{SPACE}krijgt{S
    PACE}nu{SPACE}de{SPACE}tweede{2xSP
    ACE}={SPACE}";a$
130 b$=r$:print "tweede{SPACE}krijgt{S
    PACE}nu{SPACE}de{SPACE}reserve{SPA
    CE}={SPACE}";b$
140 print
150 print "de{SPACE}computer{SPACE}maa
    kt{SPACE}er{SPACE}van{SPACE}";a$;"
    {SPACE}en{SPACE}";b$
160 end
```

Poke zoeker

Het volgende programma is geschreven door Stephan en Stefan, het is een handig programma om pokes te zoeken.

```
 10 rem *** poke zoeker ***
 20 rem door stefan potters en
 30 rem stephan vijfwinkel
 40 rem uit zwijndrecht
 60 rem (078)-100794 & (078)-101045
 70 rem ***************************
 80 print"{SHIFT CLR}{CTRL 6}":poke532
    81,0:poke53280,0
 90 let q=0:let r=0
100 print"{SHIFT CLR}{3xSPACE}programm
    eer{SPACE}deze{SPACE}poke's{SPACE}
    als{SPACE}volgt{SPACE}:{4xSPACE}"
110 print"{4xSPACE}poke,{SPACE}positie
    nummer,{SPACE}pokegetal"
120 geta$:ifa$"{SPACE}" then120
130 print"{SHIFT CLR}CCCCCCCCCCCCCCCCC
    CCCCCCCCCCCCCCCCCCCCCCC"
135 for k=1 to 255
140 print"pokenummer{SPACE}:";r;"{6xSP
    ACE}teken{SPACE}:":poke1136,q
150 print"{CRSR-DOWN}CCCCCCCCCCCCCCCCC
    CCCCCCCCCCCCCCCCCCCCCCC"
160 get a$:ifa$"{SPACE}"then160
170 let r=r+1:letq=q+1
180 print"{5xCRSR-UP}":next k:goto90
```

Border lines

Het volgende programma is van Benno Goede. Voor dit programma heeft men een Power cardridge nodig i.v.m. regel 40. Als men in regel 30 het getal 23 verandert in 19, zal de lijn langzamer lopen.

```
10 for i=49152to49169
20 read a:pokei,a:nexti
30 sys49152:poke53265,23
40 sys$c009
50 goto 30
60 data169,0,141,32,208
70 data 141,33,208,96,169
80 data 1,141,32,208,141
90 data 33,208,96
91 rem dit programma is gemaakt door
   benno goede
92 rem ruinelaan 31 bergen (n-h)
93 rem tel 02208-13182
```

*** AMIGA - COMMODORE 16 - COMMODORE 64 - COMMODORE 128 ***

Prijsvraag

voorwaarden:

° maximale lengte van het programma 200 regels.

° Alleen orginele programma's mogen worden ingezonden, de inzender blijft zelf verantwoordelijk !

° inzending alleen op een magnetisch medium dus een cassette of diskette

° geen inzendingen die u ook naar andere bladen stuurt of heeft -- gestuurd

° inzendingen voor serieuze toepassingen mogen een wat grotere lengte hebben

U kunt op twee manieren winnen:
-1- Als u als beste inzender in uw catogorie een prijs wint, of
-2- Wanneer we uw programma goed genoeg vinden om te publiceren. In dat geval krijgt u daarvoor een vergoeding uitgekeerd.

N.B. Door inzending stemt men toe in publicatie, ook in elektronische vorm. De vergoeding bij plaatsing wordt door de redaktie bepaald. Inzending krijgen na ontvangst van hun inzending van ons een andere cassette of diskette toegestuurd, dus hou zelf een kopie van het programma.
Bovendien ontvangt iedere inzender van ons een leuke attentie

Wat mag het zijn:

Utilities, spelletjes, doe-programma's, serieuze, edukatieve toepassingen.

Prijzen

Er zijn vele mooie prijzen te winnen:
- **Een kleuren Monitor**
- **Een diskdrive**
- **Verschillende printers**
- **Hardware uitbreidingen**
- **Een groot aantal software pak ketten.**

De prijzen zullen worden uitgereikt op de volgende INFO-beurs in Amsterdam. Om iedereen in de gelegenheid te stellen zijn of haar programma in de vakantie te maken of aan te passen hebben we het uiterst inzendtermijn ruim na de vakantie periode gepland. Inzendingen voorzien van naam, adres, type computer en alle gegevens die verder van belangzijn moeten bij ons binnen zijn voor 30 september 1989.
Buiten op de envelope dient duidelijk te staan **COMMODORE PRIJSVRAAG 1989 !!!**

<u>Uiterst</u> inzendtermijn
30 SEPTEMBER 1989

Enkele tips:

° Programmeer overzichtelijk, gebruik dus rem-regels om bepaalde blokken te verduidelijken. Lukt dit niet door de lengte van het programma (om bijvoorbeeld binnen de 200 regels te blijven) geef deze uitleg er dan appart bij. Hoe meer informatie we krijgen hoe beter.

° We letten bij de beoordeling van het programma niet alleen op de inhoud, maar ook op de orginaliteit, de wijze van programmeren, de handleiding en, niet in het minst, de totale verzorging.

° Als je een gebruiksaanwijzing maakt schrijf deze dan als tekstfile (het maakt niet uit met welke tekst verwerker dit is geschreven), op de cassette of diskette, achter het programma.

° Gebruik je een cassette recorder zet de programma's dan minimaal twee maal op de cassette, bijvoorbeeld voor- en achterkant gebruiken. Dit voorkomt het niet kunnen laden van een programma.

° Schrijf je naam, adres en verdere gegevens niet alleen op de envelope, maar ook op de bijbehorende brief en diskette of cassette.

° Liefst een diskette gebruiken hierdoor heeft U minder kans dat een programma niet te lezen is, en U daardoor buiten de prijzen zou vallen, zonde van de tijd en de eventuele prijs. (Als U op disk instuurt krijgt U van ons ook een nieuwe disk retour).

Commodore 1581

In het vorige nummer van Commodore Info werd al een artikel gewijd aan de Commodore 1581 diskdrive. Het onderstaande programma voor het aanmaken van een subdirectory op diskette hoorde daar bij.

```
1000 rem"************************************
1010 rem"* Dit programma maakt een 'sub-directory' voor u
    aan, *"
1020 rem"*   en werkt in zowel de 'C-64' als de 'C-128'
    mode.  *"
1030 rem"*          (C) 1987 By Master Genius in 1987
    *"
1040 rem"***********************************"
1050 :
1060 open 15,8,15:print chr$(147)
1070 :
1080 input"Sub-directory:";sd$
1090 :     rem maximaal 8 karakters
1100 if sd$="" or len(sd$)>8 then goto 1080
1110 input"Track..:";t
1120 :     rem van 1 to 39 en 41 to 80
1130 if t=40 or t<1 or t>80 then goto 1110
1140 input"Sector.:";s
1150 :     rem van 0 to 39
1160 if s<0 or s>39 then goto 1140
1170 input"Blokken:";ab
1180 :     rem min. 120 blocks en deelbaar door 40
1190 if ab=0 or ab/40 <1 then goto 1170
1200 lb=int(ab/256):hb=ab-lb*256
1210 :     rem high and low byte
1220 print#15,"/0:"+sd$+","+
   chr$(t)+chr$(s)+chr$(lb)+chr$(hb)+",c"
1230 close15
```

Checksum C-64

Syntax Checksum

Het overtikken van een listing kan een heel karwei zijn en als u een beetje normaal mens bent dan maakt u daarin beslist een aantal fouten. Nu is niets moeilijker om de fouten uit je eigen werk te halen. Al geruime tijd heeft Jan Bodzinga hiervoor een zgn. Checksum-programma geschreven. Om de vele nieuwe lezers van Commodore-info te helpen volgt hieronder nog een keer een volledige uitleg over de werking van dit programma, waarmee het, hoe vreemd dat misschien ook lijkt, echt mogelijk is om met behulp van dit programma de fouten in elke door ons geplaatste listing op te sporen.

Hiervoor gaat u als volgt te werk:

1. U tikt de listing heel zorgvuldig over en SAVEt hem voordat u het programma RUNt op een diskette of cassette.
2. U tikt het RUN commando in. Mocht het programma de boodschap 'FOUT in dataregels!' geven dan heeft u een fout bij het overtikken gemaakt. Herstel dan de fout en SAVE de verbeterde versie. Mocht het programma met de boodschap 'data is weggezet checksum testen met sys...'komen dan is tot dusver alles goed. Het programma is nu in een stukje machinetaalgeheugen gezet. Als u het NEW commando geeft blijft het toch in de computer staan.

Alle door ons geplaatste programma's zijn in Basic geschreven.

Als u een programma heeft overgetikt SAVE het eerst, mocht er iets mis gaan dan hoeft u niet de gehele listing opnieuw te gaan intikken. Als u nu een programma op fouten wilt gaan controleren dan kunt u dat in het geheugen laden(wel eerst het checksum programma hebben gerund). Vervolgens typt u zonder het programma te runnen de opdracht sys 49152(c-64) of sys 1536 (c-16 en plus/4)in.

Als alles goed is gegaan loopt er nu een rij regelnummers over het scherm met getallen erachter. Dezelfde lijst staat ook achter elk door ons geplaatste programma. Wijkt nu een nummer achter een regelnummer af van het nummer dat in het blad staat dan heeft u in die regel iets anders ingetikt dan er in het blad stond. U kunt de stroom getallen d.m.v. de RUN/STOP toets pauzeren en weer vervolgen met de F1 of F7 toets.

Het is uitermate belangrijk dat u goed met dit programma overweg kunt en mocht u het niet goed werkend krijgen bel dan gerust even met onze listingservice telefoonlijn. (Maandag 17.00 - 21.00 uur. Telefoonnummer 02155-25162.)

De laatste tijd wordt er weer veel gebeld zodat U nogwel eens in gesprek krijgt, daarom houdt uw vraag kort, vermeld in welk blad het desbetreffende artikel stond. Heeft U veel vragen, of zis uw vraag erg uitgebreid, doe het dan schriftelijk, zodat we veel mensen op de maandag avond te woord kunnen staan.

```
1     rem ******************************
2     rem basic loader "SYNTAX.CHECKSUM"
3     rem na de commando's "run" en "new"
4     rem blijft dit programma in het ge-
5     rem heugen. laad het te testen pro-
6     rem gramma en tik daarna sys 49152.
7     rem ******************************
10    i=49152  :rem beginadres
20    reada:ifa<0then40:rem data ingelezen
30    pokei,a:i=i+1:b=b+a:goto20
40    if b<>16844thenprint"[SHIFT-CLR]fo
      ut[SPACE]in[SPACE]dataregels!":b=0
      :end
50    poke49184,148:poke49185,192
55    i=49300
60    read a: ifa<0then80
70    pokei,a:b=a+b:i=i+1:goto60
80    if b<>20068thenprint"[SHIFT-CLR]fo
      ut[SPACE]in[SPACE]dataregels![SPAC
      E](vanaf[SPACE]regel[SPACE]240)":b
      =0:end
90    print"data[SPACE]is[SPACE]weggezet"
95    print"checksum[SPACE]testen[SPACE]
      met[SPACE]sys49152"
100   data 165,43,166,44,133,163,134,164
      ,169,147
110   data 32,210,255,160,0,240,3,32,73,
      192
120   data 32,73,192,208,1,96,32,225,255
      ,208
130   data 3,76,116,164,32,81,192,32,73,
      192
140   data 240,12,201,32,240,247,24,101,
      167,133
150   data 167,76,37,192,166,167,169,0,1
      32,168
160   data 32,205,189,169,13,32,210,255,
      164,168
170   data 76,17,192,200,208,2,230,164,1
      77,163
180   data 96,162,0,189,123,192,240,6,32
      ,210
190   data 255,232,208,245,32,73,192,170
      ,32,73
200   data 192,132,168,32,205,189,162,3,
      169,32
210   data 32,210,255,202,208,250,169,0,
      133,167
220   data 164,168,96,82,69,71,69,76,32,0
230   data  -1
240   data 165,197,201,3,240,7,201,4,240
250   data 6,76,148,192,76,34,192,169
260   data 147,32,210,255,76,161,192
270   data -1
```

** EINDE LISTING checksum 64 **

Checksum Checksum 64

REGEL	1	249	REGEL	60	192	REGEL	180	223
REGEL	2	84	REGEL	70	42	REGEL	190	73
REGEL	3	105	REGEL	80	244	REGEL	200	79
REGEL	4	2	REGEL	90	245	REGEL	210	109
REGEL	5	246	REGEL	95	237	REGEL	220	106
REGEL	6	152	REGEL	100	183	REGEL	230	225
REGEL	7	249	REGEL	110	158	REGEL	240	16
REGEL	10	157	REGEL	120	232	REGEL	250	163
REGEL	20	64	REGEL	130	183	REGEL	260	92
REGEL	30	38	REGEL	140	96	REGEL	270	22
REGEL	40	57	REGEL	150	96			
REGEL	50	14	REGEL	160	127			
REGEL	55	251	REGEL	170	71			

PRINT OUT C-64 met o.a. Zeeslag

Databen C64

Databen is een uitgebreide database, er zijn in de afgelopen tijden al verschillende versies van DB programma's geplaatst maar iedere keer net toch weer anders van opzet. Ook deze keer weer een versie, en was een van de vorige niet helemaal wat u zocht misschien dat deze beter voor uw doeleinden geschikt is. De inzender is de Heer A.P.W. Baelde uit Maarssen.

```
1     rem databen(k)/ben baelde/maarssen
10    poke53280,0:poke53281,0:dima$(150,10),t$(13),l(10):fori=0to12:readt$(i):next
20    l$=chr$(10)
30    t%=0:gosub1740:input"[7xCRSR-DOWN]hoofdletters[SPACE](j/n)";j$:ifj$<>"j"andj$<>"n"then30
40    ifj$="j"thenprintchr$(142):goto60
50    printchr$(14):h%=1
60    t%=0:gosub1740:printtab(16)"m[SPACE]e[SPACE]n[SPACE]u[CRSR-DOWN][7xCRSR-LEFT][7xCOM-Y]
70    printtab(4)"[CTRL-9][SPACE]1[SPACE][CTRL-0][SPACE]bestand[SPACE]maken
80    printtab(4)"[CRSR-DOWN][CTRL-9][SPACE]2[SPACE][CTRL-0][SPACE]bestand[SPACE]bekijken/wijzigen
90    printtab(4)"[CRSR-DOWN][CTRL-9][SPACE]3[SPACE][CTRL-0][SPACE]bestand[SPACE]opslaan/laden":printtab(4)"[CRSR-DOWN][CTRL-9][SPACE]4[SPACE][CTRL-0][SPACE]bestand[SPACE]printen
100   printtab(4)"[CRSR-DOWN][CTRL-9][SPACE]5[SPACE][CTRL-0][SPACE]bestand[SPACE]sorteren":printtab(4)"[CRSR-DOWN][CTRL-9][SPACE]6[SPACE][CTRL 0][SPACE]toevoegen
110   printtab(4)"[CRSR-DOWN][CTRL-9][SPACE]7[SPACE][CTRL-0][SPACE]zoeken":printtab(4)"[CRSR-DOWN][CTRL-9][SPACE]8[SPACE][CTRL-0][SPACE]diversen
120   printtab(4)"[CRSR-DOWN][CTRL-9][SPACE]9[SPACE][CTRL-0][SPACE]einde
130   printtab(4)"[CRSR-DOWN]wat[SPACE]is[SPACE]uw[SPACE]keuze";:inputk$:k=val(k$):ifk>9then60
140   if(in=0)and(k=2or(k>3andk<8))thengosub1790:goto60
150   onkgosub190,410,680,840,1550,1600,1610,1660,160:goto60
160   input"[SHIFT-CLR][6xCRSR-DOWN][4xCRSR-RIGHT]is[SPACE]het[SPACE]bestand[SPACE]gesaved[SPACE](j/n)";j$:ifj$="n"then60
170   ifj$="j"thenend
180   goto160
190   printchr$(14)
200   t%=1:gosub1740:printtab(13)"[3xCRSR-DOWN]Waarschuwing![CRSR-DOWN][13xCRSR-LEFT][13xCOM-Y]
210   print"[2xCRSR-DOWN]Bij[SPACE]het[SPACE]maken[SPACE]van[SPACE]een[SPACE]nieuw[SPACE]bestand[SPACE]wor-den[SPACE]alle[SPACE]nog[SPACE]in[SPACE]de[SPACE]";
220   print"computer[SPACE]aanwezige[3xSPACE]gegevens[SPACE]vernietigd!
230   print"[2xCRSR-DOWN][CTRL-9][SPACE]F1[SPACE][CTRL-0][SPACE]Menu":print"[CRSR-DOWN][CTRL-9][SPACE]F3[SPACE][CTRL-0][SPACE]Doorgaan
240   getg$:ifg$<>"[F1]"andg$<>"[F3]"then240
250   ifh%=1thenprintchr$(14):goto270
260   printchr$(142)
270   ifg$="[F1]"thenreturn
280   t%=2:in=0:gosub1740:gosub1780:d=1:gosub2040
290   input"[CRSR-DOWN]aantal[SPACE]velden[SPACE](max.[SPACE]10)";v$:v=val(v$):ifv=0orv>10then290
300   print"[CRSR-DOWN]datum:"d$:print"[CRSR-DOWN]bestandsnaam:"t$(13):print"[CRSR-DOWN]aantal[SPACE]velden:"v
310   gosub1760:ifj$="n"then280
320   gosub1740:fori=1tov:print"[CRSR-DOWN]naam[SPACE]veld[SPACE]"i;:inputv$(i):v$(i)=v$(i)+":":next
330   gosub1760:ifj$="n"then320
340   gosub1740:in=in+1:ifin>400thengosub1910:in=in-1:return
350   print"[CRSR-DOWN][CTRL-9][SPACE]_[SPACE][CTRL-0][SPACE]menu"
360   print"[CRSR-DOWN]kaart[SPACE]nr."in"[CRSR-DOWN]":fori=1tov:print"[CTRL 9]"v$(i)"[CTRL-0]";:inputa$(in,i)
370   ifa$(in,i)=""thena$(in,i)="*"
380   ifa$(in,i)="_"theni=v:in=in-1:next:return
390   next:gosub1760:ifj$="n"thenin=in-1
400   goto340
410   a=1
420   t%=3:gosub1740
430   print"[CTRL-9][SPACE]f1[SPACE][CTRL 0][SPACE]menu"tab(20)"[CTRL-9][SPACE]f5[SPACE][CTRL-0][SPACE]vorige[SPACE]kaart":print"[CRSR-DOWN][CTRL 9][SPACE]f3[SPACE][CTRL-0][SPACE]wijzigen";
440   printtab(20)"[CTRL-9][SPACE]f7[SPACE][CTRL-0][SPACE]volgende[SPACE]kaart":print"[CRSR-DOWN][CTRL-9][SPACE]spatiebalk[SPACE][CTRL-0][SPACE]verwijderen[CRSR-DOWN]
450   fori=1to40:print"*";:next:print"[CRSR-UP]"tab(20-(len(t$(13)+d$))/2)t$(13)"*"d$
460   print"kaart[SPACE]nr."a"[2xCRSR-DOWN]":forj=1tov:print"[CTRL-9]"v$(j)"[CTRL-0][SPACE]"a$(a,j):next
470   getg$:ifg$<>"[F1]"andg$<>"[F3]"andg$<>"[F5]"andg$<>"[F7]"andg$<>"[SPACE]"then470
480   ifg$="[F1]"thenreturn
490   ifg$="[F5]"thena=a-1:ifa<1thena=1
500   ifg$="[F7]"thena=a+1:ifa>in thena=in
510   ifg$="[F3]"thengosub540
520   ifg$="[SPACE]"thengosub640
530   goto420
540   t%=6:gosub1740:print"[CRSR-DOWN][CTRL 9][SPACE]f1[SPACE][CTRL-0][SPA
```

```
     CE]terug[SPACE]naar[SPACE]kaart"a
550  print"[CRSR-DOWN][CTRL-9][SPACE]f3
     [SPACE][CTRL-0][SPACE]veldnaam[SPA
     CE]wijzigen":print"[CRSR-DOWN][CTR
     L 9][SPACE]f5[SPACE][CTRL-0][SPACE
     ]gegevens[SPACE]wijzigen
560  print"[CRSR-DOWN][CTRL-9][SPACE]f7
     [SPACE][CTRL-0][SPACE]bestandsnaam
     [SPACE]wijzigen
570  getg$:ifg$<>"[F1]"andg$<>"[F3]"and
     g$<>"[F5]"andg$<>"[F7]"then570
580  ifg$="[F1]"thenreturn
590  ifg$="[F7]"thengosub2040
600  input"[CRSR-DOWN]welk[SPACE]veldnr
     .";i$:i1=val(i$):ifi1<1ori1>v then
     600
610  ifg$="[F3]"thenprint"[CRSR-DOWN]na
     am[SPACE]veld"i1;:inputv$(i1):v$(i
     1)=v$(i1)+":"
620  ifg$="[F5]"thenprint"[CRSR-DOWN]ge
     gevens[SPACE]veld"i1;:inputa$(a,i1)
630  return
640  input"[CRSR-DOWN]weet[SPACE]u[SPAC
     E]het[SPACE]zeker[SPACE](j/n)";j$:
     ifj$<>"j"andj$<>"n"then640
650  ifj$="n"thenreturn
660  t%=5:gosub1740:printtab(10)"[4xCRS
     R-DOWN]";:gosub1800
670  fork=atoin-1:forj=1tov:a$(k,j)=a$(
     k+1,j):nextj,k:in=in-1:a=a-1:return
680  t%=4:gosub1740:print"[CRSR-DOWN][C
     TRL 9][SPACE]f1[SPACE][CTRL-0][SPA
     CE]menu":print"[CRSR-DOWN][CTRL-9]
     [SPACE]f3[SPACE][CTRL-0][SPACE]ops
     laan
690  print"[CRSR-DOWN][CTRL-9][SPACE]f5
     [SPACE][CTRL-0][SPACE]laden":gosub
     2020:ifg$="[F1]"thenreturn
700  ifg$="[F3]"andin=0thengosub1790:re
     turn
710  printtab(3)"[2xCRSR-DOWN]staat[SPA
     CE]de[SPACE]cassetterecorder[SPACE
     ]klaar?
720  printtab(8)"[2xCRSR-DOWN]druk[SPAC
     E]dan[SPACE]op[SPACE]een[SPACE]toe
     ts
730  getg1$:ifg1$=""then730
740  ifg$="[F5]"then800
750  ifd=1thend=0:goto770
760  gosub1780
770  printtab(3)"[2xCRSR-DOWN]druk[SPAC
     E]de[SPACE]record-[SPACE]en[SPACE]
     play-toets[SPACE]in":wait1,32,32:o
     pen1,1,1
780  print#1,d$:print#1,t$(13):print#1,
     in:print#1,v:fori=1tov:print#1,v$(
     i):next
790  print"[SHIFT-CLR]":fori=1toin:forj
     =1tov:print#1,a$(i,j):nextj,i:clos
     e1:return
800  printtab(8)"[2xCRSR-DOWN]druk[SPAC
     E]de[SPACE]play-toets[SPACE]in":wa
     it1,32,32:open1,1:print"[SHIFT-CLR]
810  input#1,d$:input#1,t$(13):input#1,
     in:input#1,v:fori=1tov:input#1,v$(
     i)
820  v$(i)=v$(i)+":":next
830  fori=1toin:forj=1tov:input#1,a$(i,
     j):nextj,i:close1:return
840  ife%=1then930
850  t%=7:gosub1740:print"[CRSR-DOWN][C
     TRL 9][SPACE]f1[SPACE][CTRL-0][SPA
     CE]menu
860  print"[3xCRSR-DOWN]kijk[SPACE]of[S
     PACE]de[SPACE]printer[SPACE]klaar[
     SPACE]staat[10xSPACE]en[SPACE]druk
     [SPACE]dan[SPACE]op[SPACE]een[SPAC
     E]toets.
870  getg$:ifg$=""then870
880  ifg$="[F1]"thenreturn
890  t%=7:gosub1740:print"[CRSR-DOWN][C
     TRL 9][SPACE]f1[SPACE][CTRL-0][SPA
     CE]alle[SPACE]kaarten[SPACE]onder[
     SPACE]elkaar
900  print"[CRSR-DOWN][CTRL-9][SPACE]f3
     [SPACE][CTRL-0][SPACE]twee[SPACE]k
     aarten[SPACE]naast[SPACE]elkaar":p
     rint"[CRSR-DOWN][CTRL-9][SPACE]f5[
     SPACE][CTRL-0][SPACE]kolommen
910  print"[CRSR-DOWN][CTRL-9][SPACE]f7
     [SPACE][CTRL-0][SPACE]etiketten
920  getg1$:ifg1$<>"[F1]"andg1$<>"[F3]"
     andg1$<>"[F5]"andg1$<>"[F7]"then920
930  e%=1:t%=7:gosub1740:print"[CRSR-DO
     WN][CTRL-9][SPACE]f1[SPACE][CTRL-0
     ][SPACE]menu":print"[CRSR-DOWN][CT
     RL 9][SPACE]f3[SPACE][CTRL-0][SPAC
     E]alle[SPACE]kaarten
940  print"[CRSR-DOWN][CTRL-9][SPACE]f5
     [SPACE][CTRL-0][SPACE]bepaalde[SPA
     CE]kaarten[SPACE](nummer)
950  print"[CRSR-DOWN][CTRL-9][SPACE]f7
     [SPACE][CTRL-0][SPACE]bepaalde[SPA
     CE]kaarten[SPACE](veldgegevfn)
960  getg2$:ifg2$<>"[F1]"andg2$<>"[F3]"
     andg2$<>"[F5]"andg2$<>"[F7]"then960
970  ifg2$="[F1]"thenreturn
980  ifg2$="[F3]"thengosub1120:return
990  ifg2$="[F5]"then1010
1000 ifg2$="[F7]"thengosub1850:return
1010 t%=7:gosub1740:gosub1820:print"[CR
     SR-DOWN]voer[SPACE]de[SPACE]gewens
     te[SPACE]nummers[SPACE]in[SPACE](m
     ax.[SPACE]10)
1020 print"[CRSR-DOWN]druk[SPACE]na[SPA
     CE]elk[SPACE]nummer[SPACE]op[SPACE
     ]return[SPACE]en[SPACE]geef[SPACE]
     alslaatste[SPACE]cijfer[SPACE]een[
     SPACE]0.[CRSR-DOWN]
1030 gosub2010:fort=0to9
1040 inputi$:j%=val(i$):ifj%>in thenpri
     nt"te[SPACE]hoog[SPACE]nummer!":go
     sub1810:goto1040
1050 ifj%=0thent=9:goto1070
1060 p%(t)=j%
1070 next:gosub1760:ifj$="n"then1010
1080 gosub1120
1090 input"[CRSR-DOWN]meer[SPACE]kaarte
     n[SPACE](j/n)";j$:ifj$<>"j"andj$<>
     "n"then1090
1100 ifj$="j"thenx%=1:goto1010
1110 return
1120 ifw%=0org1$="[F5]"thengosub1800
1130 ifg1$<>"[F5]"then1180
1140 fori=1tov-1:l(i)=0:next
1150 fori=1toin:forj=1tov:iflen(a$(i,j)
```

```
     )>l(j)thenl(j)=len(a$(i,j)):nextj,i
1160 nextj,i:fori=1tov:iflen(v$(i))>l(i
     )thenl(i)=len(v$(i))
1170 next
1180 ifh%=1thenopen1,4,7:goto1200
1190 open1,4
1200 t=0:tt=0:ifg1$="[F7]"then1230
1210 t=0:tt=0
1220 ifx%=0thenprint#1,chr$(14)chr$(18)
     "[SPACE]"t$(13)chr$(146)chr$(15)"[
     SPACE]d.d.[SPACE]"d$l$
1230 fori=1toin:g=asc(g1$):ong-132gosub
     1310,1240,1280,1340:next:goto1540
1240 ifg2$="[F3]"thengosub1380:return
1250 fort=0to9:ifi=p%(t)andtt>0thenp%(t
     )=0:gosub1490:tt=0:t=9:goto1270
1260 ifi=p%(t)thenp%(t)=0:tt=i:ifp%(t+1
     )=0thengosub1370:t=9
1270 next:return
1280 ifg2$="[F3]"thengosub1420:return
1290 fort=0to9:ifi=p%(t)thenp%(t)=0:gos
     ub1420:t=9
1300 next:return
1310 ifg2$="[F3]"thengosub1370:return
1320 fort=0to9:ifi=p%(t)thenp%(t)=0:gos
     ub1370:t=9
1330 next:return
1340 ifg2$="[F3]"thengosub1530:return
1350 fort=0to9:ifi=p%(t)thenp%(t)=0:gos
     ub1530:t=9
1360 next:return
1370 gosub1510:forj=1tov:print#1,a$(i,j
     ):next:print#1,l$:return
1380 ifi+1<in thengosub1520:goto1400
1390 goto1370
1400 forj=1tov:print#1,a$(i,j);:ifi<in
     thenprint#1,chr$(16)"40"a$(i+1,j)
1410 next:i=i+1:print#1,l$:return
1420 ifz=0thengosub2060:j=1:gosub2070:p
     rint#1,chr$(16)"05"left$(v$(1),y%);
1430 ifz=0thenforj=2tov:z%=y%:gosub2070
1440 ifz=0thenprint#1,spc(l(j-1)-len(le
     ft$(v$(j-1),z%))+1)left$(v$(j),y%)
     ;:next
1450 ifz=0thengosub2060:forx=1to80:prin
     t#1,"-";:next:gosub2060:z=1
1460 print#1,i;chr$(16)"05"a$(i,1);
1470 forj=2tov:print#1,spc(l(j-1)-len(a
     $(i,j-1))+1)a$(i,j);:next
1480 print#1,chr$(8)l$chr$(15):return
1490 print#1,chr$(18)"[SPACE]kaart[SPAC
     E]nr."tt;chr$(16)"40[SPACE]kaart[S
     PACE]nr."i;chr$(146)
1500 forj=1tov:print#1,a$(tt,j)chr$(16)
     "40"a$(i,j):next:print#1,l$:return
1510 print#1,chr$(18)"[SPACE]kaart[SPAC
     E]nr."i;chr$(146):return
1520 print#1,chr$(18)"[SPACE]kaart[SPAC
     E]nr."i;chr$(16)"40[SPACE]kaart[SP
     ACE]nr."i+1;chr$(146):return
1530 forj=1to8:print#1,a$(i,j):next:pri
     nt#1:return
1540 close1:e%=0:w%=0:z=0:x%=0:return
1550 t%=8:gosub1740
1560 input"[CRSR-DOWN]op[SPACE]welk[SPA
     CE]veld[SPACE]wilt[SPACE]u[SPACE]s
     orteren";i$:s%=val(i$):ifs%<1ors%>
     vthen1560
1570 gosub1800:fori=1toin-1:forj=i+1toi
     n:ifa$(i,s%)>a$(j,s%)thengosub1590
1580 nextj,i:return
1590 forh=1tov:t$=a$(i,h):a$(i,h)=a$(j,
     h):a$(j,h)=t$:next:return
1600 t%=9:gosub340:return
1610 t%=10:gosub1740:print"[CRSR-DOWN][
     CTRL-9][SPACE]f1[SPACE][CTRL-0][SP
     ACE]menu":print"[CRSR-DOWN][CTRL-9
     ][SPACE]f3[SPACE][CTRL-0][SPACE]ee
     n[SPACE]bepaalde[SPACE]kaart
1620 print"[CRSR-DOWN][CTRL-9][SPACE]f5
     [SPACE][CTRL-0][SPACE]kaarten[SPAC
     E]met[SPACE]dezelfde[SPACE]veldgeg
     evens":gosub2020
1630 ifg$="[F1]"thenreturn
1640 ifg$="[F3]"thengosub1820:gosub420:
     return
1650 q=1:gosub1850:return
1660 t%=12:gosub1740:print"[CRSR-DOWN][
     CTRL-9][CRSR-DOWN][SPACE]f1[SPACE]
     [CTRL-0][SPACE]menu":print"[CTRL-9
     ][3xCRSR-DOWN]"t$(13)
1670 print"[CTRL-9][CRSR-DOWN]max.[SPAC
     E]aantal[SPACE]kaarten[7xSPACE]:[C
     TRL 0][SPACE]400
1680 print"[CTRL-9][CRSR-DOWN]aantal[SP
     ACE]ingevoerde[SPACE]kaarten[SPACE
     ]:[CTRL-0]"in
1690 print"[CTRL-9][CRSR-DOWN]aantal[SP
     ACE]beschikbare[SPACE]kaarten:[CTR
     L 0]"400-in
1700 print"[CTRL-9][CRSR-DOWN]aantal[SP
     ACE]velden[SPACE]per[SPACE]kaart[3
     xSPACE]:[CTRL-0]"v
1710 print"[CTRL-9][CRSR-DOWN]aantal[SP
     ACE]vrije[SPACE]bytes[8xSPACE]:[CT
     RL 0]"fre(0)
1720 getg$:ifg$<>"[F1]"then1720
1730 return
1740 print"[SHIFT-CLR]";:form=1to40:pri
     nt"#";:next:print"[HOME]"tab(20-le
     n(t$(t%))/2)t$(t%)
1750 return
1760 input"[2xCRSR-DOWN]ok[SPACE](j/n)"
     ;j$:ifj$<>"j"andj$<>"n"then1760
1770 return
1780 print"[CRSR-DOWN]wat[SPACE]is[SPAC
     E]de[SPACE]datum[SPACE]van[SPACE]v
     andaag?[CRSR-DOWN]":inputd$:d$="[S
     PACE]"+d$+"[SPACE]":return
1790 print"[SHIFT-CLR][4xCRSR-RIGHT][4x
     CRSR-DOWN]er[SPACE]is[SPACE]nog[SP
     ACE]geen[SPACE]bestand[SPACE]aanwe
     zig!":gosub1810:return
1800 print"[CRSR-DOWN]even[SPACE]geduld
     [SPACE]a.u.b."
1810 forw=1to2500:next:return
1820 print"[CRSR-DOWN]er[SPACE]zijn"in"
     kaarten[SPACE]in[SPACE]dit[SPACE]b
     estand.":ift%=7thenreturn
1830 input"[CRSR-DOWN]welk[SPACE]kaartn
     ummer[SPACE]kiest[SPACE]u";i$:a=va
     l(i$):ifa=0ora>in then1830
1840 return
1850 q=1:print"[CRSR-DOWN]welk[SPACE]ve
     ldgegeven[SPACE]zoekt[SPACE]u?[CRS
     R-DOWN]":inputi$:v%=len(i$):w%=1:g
```

```
     osub1800
1860 gosub2010:fori=qtoin:forj=1tov:fork=1tolen(a$(i,j))
1870 ifi$=mid$(a$(i,j),k,v%)thenk=len(a$(i,j)):j=v:ng=1:gosub1930
1880 nextk,j,i:iff1=1thenf1=0:ng=0:return
1890 ifng=0thenprint"[CRSR-DOWN]dit[SPACE]komt[SPACE]op[SPACE]geen[SPACE]enkele[SPACE]kaart[SPACE]voor!":goto1920
1900 ifp%(0)>0andt<10thengosub1120:return
1910 print"[CRSR-DOWN]meer[SPACE]kaarten[SPACE]zijn[SPACE]er[SPACE]niet!":
1920 ng=0:gosub1810:return
1930 ift%=7thengosub1990:return
1940 t%=11:gosub1740:print"[CRSR-DOWN][CTRL-9][SPACE]f1[SPACE][CTRL-0][SPACE]menu":print"[CRSR-DOWN][CTRL-9][SPACE]f3[SPACE][CTRL-0][SPACE]volgende[SPACE]kaart
1950 print"[CRSR-DOWN]kaart[SPACE]nr."i"[2xCRSR-DOWN]":forh=1tov:print"[CTRL 9]"v$(h)"[CTRL-0][SPACE]"a$(i,h):next
1960 getg$:ifg$<>"[F1]"andg$<>"[F3]"then1960
1970 ifg$="[F1]"theni=in:f1=1
1980 return
1990 q=i:p%(t)=i:t=t+1:ift>9thent=0:gosub1120:goto2010
2000 return
2010 fort=0to9:p%(t)=0:next:t=0:return
2020 getg$:ifg$<>"[F1]"andg$<>"[F3]"andg$<>"[F5]"then2020
2030 return
2040 print"[CRSR-DOWN]bestandsnaam[SPACE](max.[SPACE]16[SPACE]letters)[CRSR-DOWN]":inputt$(13):iflen(t$(13))>16then2040
2050 t$(13)="[SPACE]"+t$(13)+"[SPACE]":return
2060 print#1,chr$(8)1$chr$(15):return
2070 fory=1tolen(v$(j))
2080 ifmid$(v$(j),y,1)="."ormid$(v$(j),y,1)=":"theny%=y-1:y=len(v$(j))
2090 next:return
2100 data"[SPACE]databen(k)[SPACE]","[SPACE]Bestand[SPACE]maken[SPACE]","[SPACE]bestand[SPACE]maken[SPACE]"
2110 data"[SPACE]bestand[SPACE]bekijken/wijzigen[SPACE]","[SPACE]bestand[SPACE]opslaan/laden[SPACE]"
2120 data"[SPACE]verwijderen[SPACE]","[SPACE]wijzigen[SPACE]","[SPACE]bestand[SPACE]printen[SPACE]","[SPACE]bestand[SPACE]sorteren[SPACE]"
2130 data"[SPACE]toevoegen[SPACE]","[SPACE]zoeken[SPACE]","[SPACE]kaarten[SPACE]met[SPACE]dezelfde[SPACE]veldgegevens[SPACE]"
2140 data"[SPACE]diversen[SPACE]"
```

** EINDE LISTING databen **

Checksum Databen

regel 1	116	regel 440	211	regel 880	166	regel 1320	28	regel 1760	243
regel 10	189	regel 450	244	regel 890	98	regel 1330	74	regel 1770	142
regel 20	155	regel 460	228	regel 900	244	regel 1340	105	regel 1780	40
regel 30	254	regel 470	72	regel 910	154	regel 1350	26	regel 1790	47
regel 40	81	regel 480	166	regel 920	123	regel 1360	74	regel 1800	51
regel 50	160	regel 490	223	regel 930	59	regel 1370	253	regel 1810	170
regel 60	168	regel 500	169	regel 940	68	regel 1380	125	regel 1820	164
regel 70	253	regel 510	63	regel 950	159	regel 1390	84	regel 1830	21
regel 80	124	regel 520	186	regel 960	132	regel 1400	242	regel 1840	142
regel 90	102	regel 530	31	regel 970	216	regel 1410	66	regel 1850	42
regel 100	198	regel 540	148	regel 980	100	regel 1420	11	regel 1860	138
regel 110	215	regel 550	154	regel 990	14	regel 1430	144	regel 1870	87
regel 120	14	regel 560	197	regel 1000	112	regel 1440	19	regel 1880	80
regel 130	210	regel 570	135	regel 1010	56	regel 1450	98	regel 1890	142
regel 140	199	regel 580	166	regel 1020	82	regel 1460	56	regel 1900	57
regel 150	25	regel 590	110	regel 1030	30	regel 1470	193	regel 1910	122
regel 160	248	regel 600	40	regel 1040	164	regel 1480	251	regel 1920	208
regel 170	96	regel 610	166	regel 1050	77	regel 1490	125	regel 1930	188
regel 180	32	regel 620	152	regel 1060	59	regel 1500	106	regel 1940	130
regel 190	22	regel 630	142	regel 1070	247	regel 1510	250	regel 1950	255
regel 200	208	regel 640	0	regel 1080	81	regel 1520	193	regel 1960	40
regel 210	64	regel 650	114	regel 1090	115	regel 1530	181	regel 1970	62
regel 220	43	regel 660	70	regel 1100	197	regel 1540	198	regel 1980	142
regel 230	189	regel 670	242	regel 1110	142	regel 1550	246	regel 1990	251
regel 240	238	regel 680	155	regel 1120	175	regel 1560	48	regel 2000	142
regel 250	244	regel 690	220	regel 1130	199	regel 1570	42	regel 2010	190
regel 260	72	regel 700	103	regel 1140	65	regel 1580	9	regel 2020	101
regel 270	166	regel 710	62	regel 1150	244	regel 1590	172	regel 2030	142
regel 280	40	regel 720	240	regel 1160	13	regel 1600	138	regel 2040	214
regel 290	237	regel 730	213	regel 1170	130	regel 1610	51	regel 2050	176
regel 300	50	regel 740	178	regel 1180	155	regel 1620	70	regel 2060	251
regel 310	19	regel 750	224	regel 1190	48	regel 1630	166	regel 2070	138
regel 320	249	regel 760	93	regel 1200	70	regel 1640	150	regel 2080	45
regel 330	14	regel 770	168	regel 1210	250	regel 1650	145	regel 2090	74
regel 340	172	regel 780	228	regel 1220	38	regel 1660	51	regel 2100	140
regel 350	38	regel 790	102	regel 1230	154	regel 1670	9	regel 2110	179
regel 360	203	regel 800	89	regel 1240	108	regel 1680	2	regel 2120	87
regel 370	204	regel 810	196	regel 1250	168	regel 1690	108	regel 2130	205
regel 380	20	regel 820	190	regel 1260	52	regel 1700	235	regel 2140	3;
regel 390	241	regel 830	168	regel 1270	74	regel 1710	189		
regel 400	32	regel 840	27	regel 1280	103	regel 1720	218		
regel 410	36	regel 850	75	regel 1290	24	regel 1730	142		
regel 420	241	regel 860	17	regel 1300	74	regel 1740	73		
regel 430	228	regel 870	120	regel 1310	107	regel 1750	142		

Kubus C64

Het blijft boeien de kubus, ditmaal weer een andere variant door op de toetsen 0-9 te drukken verschuift een rij, breng alle gelijke kleuren in een rij. makkelijk he !!!! of niet soms ??? De maker van dit spelletje is onze vaste inzender uit Belgie, Dewijn Dieter.

```
100    print"[SHIFT-CLR]"
110    print"[3xSPACE]het[SPACE]5-kleuren
       [SPACE]probleem"
120    print"[3xSPACE][22xCOM-@]"
130    print"[3xSPACE]bij[SPACE]d.dewijn"
140    print
150    print
160    fori=1to5000:nexti
170    print"[SHIFT-CLR][COM-5]":poke5328
       1,1:poke36879,59
180    printtab(6)"[CTRL-9][SPACE]het[SPA
       CE]vijf[SPACE]kleuren[SPACE]proble
       em[SPACE]"
190    dima%(5,5):fori=0to4:reada%:forj=0
       to4:a%(i,j)=a%:nextj,i
200    data 155,152,151,154,28
210    rem * grijs 3,grijs 2,grijs 1,
                     licht blauw,rood *
220    goto380
230    rem * schermlayout *
240    print"[CRSR-DOWN][CTRL-1][10xCRSR-
       RIGHT]UC6CC7CC8CC9CC0CI"
250    fori=5to1step-1:printspc(10)"[SHIF
       T -]";spc(15);"[SHIFT--]"
260    a$=right$(str$(i),1):printspc(10)a
       $;spc(15);"B"
270    printspc(10)"[SHIFT--]";spc(15);"[
       SHIFT--]"
280    next:print"[10xCRSR-RIGHT]JCCCCCCC
       CCCCCCCCK"
290    print"[HOME][3xCRSR-DOWN]";:fori=0
       to4:forj=1to3:print:printspc(10)"[
       CTRL-9][CRSR-RIGHT]";:fork=0to4
300    printchr$(a%(i,k));"[3xSPACE]";:ne
       xtk,j,i
310    return
320    rem * verdraaien *
330    fori=5to1step-1:a%(a,i)=a%(a,i-1):
       next
340    a%(a,0)=a%(a,5):return
350    fori=5to1step-1:a%(i,a)=a%(i-1,a):
       next
360    a%(0,a)=a%(5,a):return
370    rem *hoofdprogramma *
380    gosub240:forw=0to20:a=int(5*rnd(1)
       ):on1+2*rnd(1)gosub330,350:gosub29
       0:nextw
390    geta$:ifa$="q"thensys65234
400    ifa$="_"then run
410    if a$="1"or a$="2"or a$="3"or a$="
       4"    or a$="5"or a$="6"then gosub
        470
420    if a$="7"or a$="8"or a$="9"or a$="
       0"                  then gosub 470
430    a=val(a$):ifa$="0"then a=10
440    ifa>5thena=a-6:gosub350:goto460
450    a=5-a:gosub330
460    gosub290:goto390
470    tel=tel+1:print"[HOME][COM-5][11xC
       RSR-DOWN]"    tab(28)"beurt[SPACE]"
       ;tel
480    if tel=1 then gosub 500
490    return
500    print tab(5)"[9xCRSR-DOWN]";
                    "voor[SPACE]volgend[SPAC
       E]spel,druk[SPACE]op[SPACE][CTRL-9
       ][SPACE]_[SPACE]"
510    print tab(7)
             "[CRSR-DOWN]om[SPACE]te[SPAC
       E]stoppen,druk[SPACE]op[SPACE][CTR
       L 9][SPACE]q[SPACE]"
520    return
```

** EINDE LISTING kubus **

Checksum Kubus c-64

regel 100	112	regel 330	105
regel 110	140	regel 340	39
regel 120	245	regel 350	105
regel 130	229	regel 360	39
regel 140	153	regel 370	249
regel 150	153	regel 380	139
regel 160	27	regel 390	192
regel 170	181	regel 400	118
regel 180	109	regel 410	33
regel 190	186	regel 420	174
regel 200	1	regel 430	185
regel 210	201	regel 440	42
regel 220	36	regel 450	113
regel 230	131	regel 460	135
regel 240	234	regel 470	240
regel 250	251	regel 480	28
regel 260	5	regel 490	142
regel 270	182	regel 500	241
regel 280	189	regel 510	179
regel 290	31	regel 520	14
regel 300	212		
regel 310	142		
regel 320	196		

Groeislang C64

Alexander ter Haar is de maker van dit programmaatje. De bedoeling is om met de steeds groter wordende slang het vraagteken op te eten. Als dat is gebeurd, verschijnt er een uitroepteken. Na ook dat opgegeten te hebben verschijnt er boven in de rand, op een willekeurige plaats, een pijltje omhoog. Na daar doorheen te zijn gegaan komt men in de tweede ronde.

```
10     goto 795
20     print"[COM-3][SPACE]"
30     c=429:e=+40:h=1:g=0
40     poke 53281,2:poke 53280,0
50     poke 55296,3
60     print"[SHIFT-CLR]"
70     print"[CTRL-1]"
80     for a=1to39:poke 1023+a,42:next
90     for a=1to 1000step 40:poke 1023+a,
       42:next a
100    for a=1to 960step 40:poke 1062+a,4
       2:next a
110    for a=1to40:poke 1983+a,42:next
120    poke 1465+int(30*rnd(1)),63
130    b=peek (56320)
140    if b=127 then goto 310
150    if b=126 then c=c-40:e=-40
160    if b=125 then c=c+40:e=+40
```

```
170   if b=123 then c=c-1:e=-1
180   if b=119 then c=c+1:e=+1
190   if b=122 then c=c-41:e=-41
200   if b=118 then c=c-39:e=-39
210   if b=117 then c=c+41:e=+41
220   if b=121 then c=c+39:e=+39
230   d=peek(1024+c)
240   if d=42 then goto 400
250   if d=81 then goto 400
260   if d=63 then poke 1925,33:poke 561
      97,5
270   if d=30then goto 600
280   if d=33 then poke 1024+int(31*rnd(
      1)),30
290   poke 1024+c,81:poke 55296+c,int(12
      *rnd(4)+1)
300   goto 130
310   c=c+e
320   d=peek(1024+c)
330   if d=42 then goto 400
340   if d=81 then goto 400
350   if d=63 then poke 1925,33
360   if d=30then goto 600
370   if d=33 then poke 1024+int(31*rnd(
      1)),30
380   poke 1024+c,81:poke 55296+c,int(10
      *rnd(0)+1)
390   goto 130
400   print"[7xCRSR-DOWN][CRSR-RIGHT][9x
      SPACE]game[SPACE]over[SPACE]!!!!"
410   for f=1to 5
420   poke 53281,3:poke 53280,2
430   for a=1to100:next a
440   poke 53281,2:poke 53280,3
450   for a=1to100:next a
460   poke 53281,3:poke 53280,2
470   for a=1to100:next a
480   poke 53281,2:poke 53280,3
490   for a=1to100:nexta
500   next  f
510   poke 53281,1:print"[SHIFT-CLR][CTR
      L 1]"
520   print"[6xCRSR-DOWN][SPACE]game[SPA
      CE]over,[SPACE]je[SPACE]hebt[SPACE]"
530   print"[6xSPACE]";g
540   print"[4xSPACE]punten"
550   print"[5xCRSR-DOWN][6xSPACE]wil[SP
      ACE]je[SPACE]nog[SPACE]een[SPACE]k
      eer?"
560   get f$:if f$=""then goto 560
570   if f$="j"then goto 795
580   if f$="n"then end
590   goto 560
600   print"[SHIFT-CLR][3xCRSR-DOWN][4xS
  PACE]je[SPACE]hebt[SPACE]ronde[SPACE]
610   print"[3xSPACE]";h
620   print"[2xSPACE]gehaald!!"
630   print"[2xCRSR-RIGHT][6xSPACE]nu[SP
      ACE]komt[SPACE]ronde[SPACE]";h+1
640   for k=1to900:nextk:print"[SHIFT-CL
      R]"
650   h=h+1:g=g+100
660   for a=1to39:poke 1023+a,42:next
670   for a=1to 1000step 40:poke 1023+a,
      42:next a
680   for a=1to 960step 40:poke 1062+a,4
      2:next a
690   for a=1to40:poke 1983+a,42:next
700   poke 1465+int(30*rnd(1)),63
710   j=h*11
720   for i=1to j
730   f=int (30*rnd(j)+1)
740   poke 1669+f,42:next i
750   for a=1to(h*4)
760   f=int (30*rnd(j)+1)
770   poke 1304+f,42:next a
780   c=429:e=-1
790   goto 130
795   poke 53281,1:poke 53280,1:print"[S
      HIFT CLR][CTRL-1]"
800   print"[SHIFT-CLR][3xCRSR-DOWN][4xC
      RSR-RIGHT][5xSPACE]groei[SPACE]slang"
810   print"[3xCRSR-DOWN][SPACE]de[SPACE
      ]bedoeling[SPACE]is[SPACE]om[SPACE
      ]met[SPACE]de[SPACE]slang"
820   print"[SPACE]het[SPACE]-?-[SPACE]t
      eken[SPACE]op[SPACE]te[SPACE]eten"
830   print"[SPACE]als[SPACE]dat[SPACE]i
      s[SPACE]gebeurt,[SPACE]verschijnd"
840   print"[SPACE]er[SPACE]ergens[SPACE
      ]op[SPACE]het[SPACE]scherm[SPACE]e
      en[SPACE]-!-[SPACE]teken[SPACE]"
850   print"[SPACE]na[SPACE]ook[SPACE]di
      e[SPACE]te[SPACE]hebben[SPACE]opge
      geten,"
860   print"[SPACE]verschijnt[SPACE]er[S
      PACE]ergens[SPACE]in[SPACE]de[SPAC
      E]rand"
870   print"[SPACE]boven[SPACE]aan[SPACE
      ]het[SPACE]beeld[SPACE]een[SPACE]p
      ijltje[SPACE]-^-"
880   print"[SPACE]je[SPACE]moet[SPACE]d
      an[SPACE]door[SPACE]dat[SPACE]pijl
      tje[SPACE]heenlopen"
890   print"[SPACE]en[SPACE]je[SPACE]kom
      t[SPACE]in[SPACE]de[SPACE]volgende
      [SPACE]rond"
900   print"[5xCRSR-DOWN][SPACE][CTRL-9]
      return[CTRL-0]"
910   get a$:if a$=""then goto 910
920   g=0
930   goto 20
```

regel 10	46	regel 330	171	regel 650	210
regel 20	115	regel 340	174	regel 660	133
regel 30	155	regel 350	139	regel 670	40
regel 40	39	regel 360	170	regel 680	9
regel 50	1	regel 370	120	regel 690	140
regel 60	112	regel 380	255	regel 700	248
regel 70	109	regel 390	29	regel 710	82
regel 80	133	regel 400	75	regel 720	155
regel 90	40	regel 410	131	regel 730	62
regel 100	9	regel 420	42	regel 740	244
regel 110	140	regel 430	215	regel 750	194
regel 120	248	regel 440	42	regel 760	62
regel 130	7	regel 450	215	regel 770	222
regel 140	221	regel 460	42	regel 780	161
regel 150	70	regel 470	215	regel 790	29
regel 160	67	regel 480	42	regel 795	97
regel 170	221	regel 490	215	regel 800	2
regel 180	224	regel 500	200	regel 810	62
regel 190	68	regel 510	49	regel 820	96
regel 200	87	regel 520	119	regel 830	92
regel 210	70	regel 530	95	regel 840	68
regel 220	79	regel 540	183	regel 850	46
regel 230	189	regel 550	207	regel 860	125
regel 240	171	regel 560	251	regel 870	6
regel 250	174	regel 570	10	regel 880	49
regel 260	201	regel 580	96	regel 890	225
regel 270	170	regel 590	36	regel 900	182
regel 280	120	regel 600	205	regel 910	240
regel 290	5	regel 610	96	regel 920	41
regel 300	29	regel 620	5	regel 930	23
regel 310	39	regel 630	203		
regel 320	189	regel 640	157		

Blocks free

Edwin Grootjes uit Utrecht is de maker van het volgende programma. Met het programma "blocks free" kan je het aantal blocks van je diskette veranderen. de werkwijze is als volgt. Stop een willekeurige diskette in de drive. Run het programma en druk op de spatiebalk. Geef nu het aantal blocks en geef aan of nu de diskette tegen validaten en data opslag beschermd moet worden. Het programma verandert nu het aantal blocks in track 18.00. RESET nu hardwarematig de drive EN de computer als je direkt de diskette wilt validaten(als je direkt validate kan het misgaan en zou het zonde zijn van de diskette) Nog een belangrijke tip van de maker: Maak altijd een veiligheidskopie van de te veranderen diskette als het onverhoopt mis mocht gaan is de kostbare data NIET verloren.

```
1     rem ******* (c) 1987 ergo-soft *******
10    poke53281,0:poke53280,0:dimg$(255):printchr$(142);chr$(8)
20    print"[SHIFT-CLR][CTRL-8][13xCRSR-DOWN][3xSPACE][COM-A]";:forr=1to31:print"[SHIFT-*]";:next:print"[COM S]"
30    forr=1to5:print"[3xSPACE]B[31xSPACE]B" :next
40    print"[3xSPACE][COM-Z][5xSHIFT-*][COM-R]CCCCCCCCCCCCCCCCC[COM-R]CCCCCC[COM-X]"
50    print"[9xCRSR-RIGHT]B[18xCRSR-RIGHT]B"
60    print"[9xCRSR-RIGHT][COM-Z]C[3xSHIFT *]CCCCCCCCCCCCCC[COM-X]"
70    print"[HOME][CTRL-3][16xCRSR-DOWN][6xCRSR-RIGHT][4xSPACE]stop[SPACE]disk[SPACE]in[SPACE]drive"
80    get a$:ifa$><"[SPACE]"then 80
90    print"[HOME][CTRL-3][16xCRSR-DOWN][6xCRSR-RIGHT][22xSPACE]"
100   gosub 420
110   print"[HOME][CTRL-3][14xCRSR-DOWN][13xCRSR-RIGHT]lees[SPACE]18.00:"
120   open15,8,15:open2,8,2,"#":print#15,"u1:2,"0,18,0
130   print"[5xCRSR-RIGHT]";:fora=0to255
140   ifa/ 9=int(a/9 )then print".";
150   a$="":get#2,a$:ifa$=""thena$=chr$(0)
160   g$(a)=a$:next
170   print"[HOME][14xCRSR-DOWN][12xCRSR-RIGHT][2xSPACE]wis[SPACE]18.00:[CTRL 8]"
180   print"[5xCRSR-RIGHT].";:forb=4to143:ifb/5=int(b/5)then print".";
190   g$(b)=chr$(0):nextb
200   input"[HOME][17xCRSR-DOWN][5xCRSR-RIGHT][CTRL-3]hoeveel[SPACE]blocks[SPACE](0-4335)";bl$
210   bl=val(bl$)
220   ifbl<0or bl>4335 then200
230   ba=int(bl/255):bb=bl-(ba*255)
240   p=4:ifbl<=255 then260
250   forw=1toba:g$(p)=chr$(255):p=p+8:next
260   g$(p+8)=chr$(bb)
270   open1,0:print"[CRSR-UP][5xCRSR-RIGHT][3xSPACE]anti[SPACE]validate[SPACE]?[SPACE](j/n)[SPACE]n[3xSPACE][4xCRSR-LEFT]";:input#1,h$
280   ifleft$(h$,1)="j"theng$(2)=chr$(66)
290   open3,8,3,"#"
300   print"[CRSR-UP][7xCRSR-RIGHT][CTRL 3]ergo[CTRL-8]soft[SPACE]it’s[SPACE]lonely[SPACE]at[SPACE]the[SPACE]top"
310   print"[HOME][CTRL-3][14xCRSR-DOWN][12xCRSR-RIGHT]schrijf[SPACE]18.00:"
320   print"[5xCRSR-RIGHT].";
330   forc=2to255
340   print#15,"b-p:"3,c
350   print#3,g$(c);
360   ifc/9=int(c/9)then print".";
370   next
380   print#15,"u2:3,"0,18,0
390   print#15,"i"
400   close1:close2:close3:close15
410   print"[HOME][14xCRSR-DOWN][4xCRSR-RIGHT]=>[2xSPACE][CTRL-8]reset[SPACE][CTRL-3]drive[SPACE]en[SPACE]computer[2xSPACE]<=[CTRL-1][HOME]"
415   end
420   open1,8,15,"i":input#1,m$,n$,o$,p$
430   print"[HOME][20xCRSR-DOWN][10xCRSR-RIGHT]status:"m$","n$","o$","p$
440   ifm$><"00"thenprinttab(13);"[CTRL-3][5xCRSR-UP][CTRL-9]disk[SPACE]fout[SPACE]![CTRL-0]":close1:goto460
450   close1:return
460   geta$:ifa$="[SPACE]" then run
470   goto460
```

** EINDE LISTING blocksfree **

Checksum Blocksfree

regel 1	222	regel 280	120
regel 10	74	regel 290	40
regel 20	97	regel 300	175
regel 30	230	regel 310	144
regel 40	179	regel 320	215
regel 50	112	regel 330	232
regel 60	249	regel 340	9
regel 70	76	regel 350	49
regel 80	231	regel 360	130
regel 90	202	regel 370	130
regel 100	35	regel 380	175
regel 110	205	regel 390	183
regel 120	66	regel 400	42
regel 130	199	regel 410	162
regel 140	126	regel 415	128
regel 150	66	regel 420	232
regel 160	208	regel 430	158
regel 170	252	regel 440	122
regel 180	168	regel 450	153
regel 190	246	regel 460	87
regel 200	218	regel 470	35
regel 210	8		
regel 220	243		
regel 230	215		
regel 240	201		
regel 250	184		
regel 260	60		
regel 270	182		

Zeeslag C64

Door het opgeven van coördinaten wordt er op de door de computer weggezette schepen geschoten. Bij ieder schot wordt aangegeven of het schot raak of mis is.(Dit laatste is meestal het geval) Wordt er geschoten op een coördinaat waar reeds eerder op geschoten is, of wordt er geschoten op een niet bestaand coördinaat, dan volgt een foutmelding. Is een schip gezonken, dan wordt dit aangegeven. Evenals het zinken van de totale vloot. Dit laatste met vermelding van het aantal benodigde beurten. T.o.v. het bord spel is hier het speelveld kleiner en is er 1 schip minder. De inzender is P.D. Pieterse uit Aagtekerke.

```
100   rem          ***************
110   rem          **           **
120   rem          **  zeeslag  **
130   rem          **           **
140   rem          ***************
150   rem
160   rem   door  p.d.pieterse
170   rem
180   rem         aagtekerke
190   rem
200   rem
210   rem
220   rem
230   rem      zwart/wit uitvoering
240   rem             voor
250   rem         commodore 64
260   rem
270   rem              *
280   rem            *   *
290   rem          *       *
300   rem      dit programma neemt
310   rem      9445 bytes in beslag.
320   print "[SHIFT-CLR]"
330   poke 53280,6 : poke 53281,6 : prin
      t chr$(14);
340   print "[CTRL-1][5xCRSR-DOWN]"
350   print "[3xSPACE][5xCOM-+][SPACE][4
      xCOM-+][SPACE][4xCOM-+][SPACE][4xC
      OM +][SPACE][COM-+][3xSPACE][4xCOM
       +][SPACE][4xCOM-+]"
360   print "[6xSPACE][COM-+][2xSPACE][C
      OM +][4xSPACE][COM-+][4xSPACE][COM
       +][4xSPACE][COM-+][3xSPACE][COM-+
      ][2xSPACE][COM-+][SPACE][COM-+]"
370   print "[5xSPACE][COM-+][3xSPACE][3
      xCOM-+][2xSPACE][3xCOM-+][2xSPACE]
      [4xCOM-+][SPACE][COM-+][3xSPACE][4
      xCOM-+][SPACE][COM-+][SPACE][2xCOM
       +]"
380   print "[4xSPACE][COM-+][4xSPACE][C
      OM +][4xSPACE][COM-+][7xSPACE][COM
       +][SPACE][COM-+][3xSPACE][COM-+][
      2xSPACE][COM-+][SPACE][COM-+][2xSP
      ACE][COM-+]"
390   print "[3xSPACE][5xCOM-+][SPACE][4
      xCOM-+][SPACE][4xCOM-+][SPACE][4xC
      OM +][SPACE][3xCOM-+][SPACE][COM-+
      ][2xSPACE][COM-+][SPACE][4xCOM-+]"
400   print "[8xCRSR-DOWN]"
410   print "[CTRL-6]"
420   print "[9xSPACE]9[SPACE]seconden[S
      PACE]geduld[SPACE]aub"
430   print "[CTRL-1]"
440   gosub 3550
450   print "[SHIFT-CLR][4xCRSR-DOWN]"
460   input "Wil[SPACE]je[SPACE]instrukt
      ies[SPACE](j/n)";a$
470   if a$ = "j" then goto 530
480   if a$ = "n" then goto 1280
490   print "[2xCRSR-DOWN]";
500   print "het[SPACE]antwoord[SPACE]mo
      et[SPACE][CTRL-9]j[CTRL-0][SPACE]o
      f[SPACE][CTRL-9]n[CTRL-0][SPACE]zi
      jn."
510   print "[2xCRSR-DOWN]";
520   goto 460
530   rem        instrukties
540   rem        ===========
550   print"[SHIFT-CLR]";
560   print chr$(14)
570   print "Bij[SPACE]zeeslag[SPACE]wor
      den[SPACE]door[SPACE]de[SPACE]comp
      uter"
580   print "ergens[SPACE]op[SPACE]het[S
      PACE]bord[SPACE]de[SPACE]volgende[
      SPACE]schepen"
590   print "weggezet"
600   print "[2xCRSR-DOWN]";
610   print "1[SPACE]Vliegdekmoederschip
      "
620   print "[2xSPACE](over[SPACE]5[SPAC
      E]vakken)
630   print "[2xCRSR-DOWN]";
640   print "1[SPACE]Kruiser"
650   print "[2xSPACE](over[SPACE]4[SPAC
      E]vakken)
660   print "[2xCRSR-DOWN]";
670   print "1[SPACE]Onderzeeboot"
680   print "[2xSPACE](over[SPACE]3[SPAC
      E]vakken)
690   print "[2xCRSR-DOWN]";
700   print "1[SPACE]Mijnenveger"
710   print "[2xSPACE](over[SPACE]2[SPAC
      E]vakken)
720   poke v+16,0
730   poke v+0,220:rem x pos. sprite 0
740   poke v+2,220:rem x pos. sprite 1
750   poke v+4,220:rem x pos. sprite 2
760   poke v+6,220:rem x pos. sprite 3
770   poke v+1,70 :rem y pos. sprite 0
780   poke v+3,100:rem y pos. sprite 1
790   poke v+5,130:rem y pos. sprite 2
800   poke v+7,170:rem y pos. sprite 3
810   poke v+21,15 :rem   insch. sprites
820   print "[2xCRSR-DOWN]";
830   input "Type[SPACE]om[SPACE]verder[
      SPACE]te[SPACE]gaan[SPACE]de[SPACE
      ][CTRL-9]RETURN[CTRL-0]-toets";a$
840   poke v+21,0
850   print "[SHIFT-CLR]"
860   print "De[SPACE]schepen[SPACE]kunn
      en[SPACE]zowel[SPACE]horizontaal[S
      PACE]als"
870   print "vertikaal[SPACE]op[SPACE]he
      t[SPACE]bord[SPACE]liggen."
880   print
890   print "De[SPACE]schepen[SPACE]kunn
      en[SPACE]niet[SPACE]in[SPACE]elkaa
```

```
         rs"
 900     print "verlengde[SPACE]vast[SPACE]
         aan[SPACE]elkaar[SPACE]liggen."
 910     print "Wel[SPACE]kunnen[SPACE]ze[S
         PACE]naast[SPACE]of[SPACE]haaks[SP
         ACE]op[SPACE]elkaar"
 920     print "aan[SPACE]elkaar[SPACE]vast
         [SPACE]liggen."
 930     print
 940     print "Het[SPACE]is[SPACE]de[SPACE
         ]bedoeling[SPACE]de[SPACE]schepen[
         SPACE]tot"
 950     print "zinken[SPACE]te[SPACE]breng
         en."
 960     print "[2xCRSR-DOWN]";
 970     print "De[SPACE]schoten[SPACE]moet
         en[SPACE]aangegeven[SPACE]worden"
 980     print "door[SPACE]het[SPACE]gewens
         te[SPACE]vak[SPACE]te[SPACE]kiezen
         ."
 990     print "Bijvoorbeeld[SPACE]C6[SPACE
         ]gevolgd[SPACE]door[SPACE][CTRL-9]
         RETURN[CTRL-0]."
1000     print "[2xCRSR-DOWN]";
1010     print "Is[SPACE]het[SPACE]schot[SP
         ACE]dan[SPACE]raak,[SPACE]dan[SPAC
         E]verschijnt"
1020     print "er[SPACE]in[SPACE]het[SPACE
         ]gekozen[SPACE]vak[SPACE][COM-+]."
1030     print "Is[SPACE]het[SPACE]schot[SP
         ACE]mis,[SPACE]dan[SPACE]verschijn
         t[SPACE]er[SPACE][CTRL-6][COM-B][C
         TRL 1]."
1040     print "[2xCRSR-DOWN]";
1050     input "Type[SPACE]om[SPACE]verder[
         SPACE]te[SPACE]gaan[SPACE]de[SPACE
         ][CTRL-9]RETURN[CTRL-0]-toets";a$
1060     print "[SHIFT-CLR]";
1070     print "Is[SPACE]een[SPACE]schip[SP
         ACE]gezonken,[SPACE]dan[SPACE]word
         t[SPACE]het"
1080     print "[CTRL-6]witte[CTRL-1][SPACE
         ]scheepje[SPACE]aan[SPACE]de[SPACE
         ]zijkant[SPACE]zwart."
1090     print"[2xCRSR-DOWN]"
1100     print "voorbeeld:"
1110     print "[2xCRSR-DOWN]";
1120     print "[5xSPACE]op[SPACE]zee[10xSP
         ACE]gezonken"
1130     print "[2xCRSR-DOWN]";
1140     print "Is[SPACE]de[SPACE]gehele[SP
         ACE]vloot[SPACE]gezonken,[SPACE]da
         n[SPACE]wordt"
1150     print "aangegeven[SPACE]in[SPACE]h
         oeveel[SPACE]schoten[SPACE]dit"
1160     print "gebeurd[SPACE]is.
1170     poke v+4,64   :rem x pos. sprite 2
1180     poke v+5,70 :rem y pos. sprite 2
1190     poke v+21,4   :rem    insch. sprites
1200     poke v+8,200:rem x pos. sprite 4
1210     poke v+9,70   :rem y pos. sprite 4
1220     poke v+21,20:rem    insch. sprites
1230     print "[3xCRSR-DOWN]"
1240     print "[CTRL-6][12xSPACE]VEEL[SHIF
         T SPACE]SUCCES[CTRL-1]"
1250     print "[2xCRSR-DOWN]"
1260     input "Type[SPACE]om[SPACE]verder[
         SPACE]te[SPACE]gaan[SPACE]de[SPACE
```

```
         ][CTRL-9]RETURN[CTRL-0]-toets";a$
1270     poke v+21,0
1280     print "[SHIFT-CLR]"
1290     print "[10xCRSR-DOWN]"
1300     print "[13xSPACE]Nog[SPACE]even[SP
         ACE]geduld."
1310     print
1320     print "[3xSPACE]Ik[SPACE]moet[SPAC
         E]de[SPACE]schepen[SPACE]nog[SPACE
         ]wegzetten."
1330     print"[CTRL-1]" ;
1340     for x= 0to 7
1350     for y = 0to 10
1360     let b$(x,y) = "."
1370     next y
1380     next x
1390     rem plaatsen vliegtuigmoederschip.
1400     rem ==============================
1410     vh =int(rnd(1)*(3-1)+1)
1420     if vh = 2 then goto 1500
1430     r =int(rnd(1)*(8-1)+1)
1440     k = int(rnd(1)*(11-5)+5)
1450     for x = 1 to 5
1460     k1=k-x+1
1470     let b$(r,k1)="vl"
1480     next
1490     goto 1560
1500     k = int(rnd(1)*(11-1)+1)
1510     r =int(rnd(1)*(8-5)+5)
1520     for x = 1 to 5
1530     r1 = r-x+1
1540     let b$(r1,k)="vl"
1550     next x
1560     rem         plaatsen kruiser.
1570     rem         ================
1580     vh =int(rnd(1)*(3-1)+1)
1590     if vh = 2 then goto 1720
1600     r =int(rnd(1)*(8-1)+1)
1610     k = int(rnd(1)*(11-4)+4)
1620     for x = k to k-4 step -1
1630     if b$(r,x) <> "." then goto 1600
1640     next x
1650     if k = 10then goto 1670
1660     if b$(r,k+1) <> "." then goto 1600
1670     for x = 1 to 4
1680     k1=k-x+1
1690     let b$(r,k1)="kr"
1700     next x
1710     goto 1830
1720     k = int(rnd(1)*(11-1)+1)
1730     r =int(rnd(1)*(8-4)+4)
1740     for x = r to r-4 step -1
1750     if b$(x,k) <> "." then goto 1720
1760     next x
1770     if r = 7 then goto 1790
1780     if b$(r+1,k) <> "." then goto 1720
1790     for x = 1 to 4
1800     r1 = r-x+1
1810     let b$(r1,k)="kr"
1820     next x
1830     rem         plaatsen onderzeeboot
1840     rem         ====================
1850     vh =int(rnd(1)*(3-1)+1)
1860     if vh = 2 then goto 1990
1870     r =int(rnd(1)*(8-1)+1)
1880     k = int(rnd(1)*(11-3)+3)
1890     for x = k to k-3 step -1
1900     if b$(r,x) <> "." then goto 1870
```

```
1910  next x
1920  if k = 10then goto 1940
1930  if b$(r,k+1) <> "." then goto 1870
1940  for x = 1 to 3
1950  k1=k-x+1
1960  let b$(r,k1)="oz"
1970  next x
1980  goto 2100
1990  k = int(rnd(1)*(11-1)+1)
2000  r =int(rnd(1)*(8-3)+3)
2010  for x = r to r-3 step -1
2020  if b$(x,k) <> "." then goto 1990
2030  next x
2040  if r = 7 then goto 2060
2050  if b$(r+1,k) <> "." then goto 1990
2060  for x = 1 to 3
2070  r1 = r-x+1
2080  let b$(r1,k)="oz"
2090  next x
2100  rem        plaatsen mijnenveger
2110  rem        ====================
2120  vh =int(rnd(1)*(3-1)+1)
2130  if vh = 2 then goto 2260
2140  r =int(rnd(1)*(8-1)+1)
2150  k = int(rnd(1)*(11-2)+2)
2160  for x = k to k-2 step -1
2170  if b$(r,x) <> "." then goto 2140
2180  next x
2190  if k = 10then goto 2210
2200  if b$(r,k+1) <> "." then goto 2140
2210  for x = 1 to 2
2220  k1=k-x+1
2230  let b$(r,k1)="mv"
2240  next x
2250  goto 2370
2260  k = int(rnd(1)*(11-1)+1)
2270  r =int(rnd(1)*(8-2)+2)
2280  for x = r to r-2 step -1
2290  if b$(x,k) <> "." then goto 2260
2300  next x
2310  if r = 7 then goto 2330
2320  if b$(r+1,k) <> "." then goto 2260
2330  for x = 1 to 2
2340  r1 = r-x+1
2350  let b$(r1,k)="mv"
2360  next x
2370  rem       bord tekenen
2380  rem       ============
2390  print chr$(142)
2400  poke v+21,0
2410  print"[SHIFT-CLR]";
2420  print "[2xSPACE]a[2xSPACE]b[2xSPAC
      E]c[2xSPACE]d[2xSPACE]e[2xSPACE]f[
      2xSPACE]g[2xSPACE]h[2xSPACE]i[2xSP
      ACE]j"
2430  q=49
2440  for y= 1 to 7
2450  print"[SPACE]";
2460  for x = 1 to 9
2470  print"O[2xCOM-Y]";
2480  nextx
2490  print"O[COM-Y]P"
2500  print chr$(q);
2510  for x = 1 to 28 step 3
2520  print tab(x)"[COM-H]";
2530  next x
2540  print"[SPACE]";"[COM-N]"
2550  print"[SPACE]";
2560  for x = 1 to 28 step 3
2570  print tab(x)"[COM-H]";
2580  next x
2590  print"[SPACE]";"[COM-N]"
2600  q=q+1
2610  next y
2620  print"[SPACE]";
2630  for x = 1 to 30
2640  print "[COM-Y]";
2650  next x
2660  print"[HOME][CRSR-DOWN]"
2670  print"[CTRL-6]";
2680  poke v+16,15
2690  poke v+0,30 :rem x pos. sprite 0
2700  poke v+2,30 :rem x pos. sprite 1
2710  poke v+4,30 :rem x pos. sprite 2
2720  poke v+6,30 :rem x pos. sprite 3
2730  poke v+1,50 :rem y pos. sprite 0
2740  poke v+3,100:rem y pos. sprite 1
2750  poke v+5,150:rem y pos. sprite 2
2760  poke v+7,200:rem y pos. sprite 3
2770  poke v+21,15 :rem   insch. sprites
2780  rem       vragen waar.
2790  rem       ============
2800  print"[HOME][23xCRSR-DOWN]";
2810  print "waar[SPACE]komt[SPACE]het[S
      PACE]schot";"[11xSPACE]";"[11xCRSR
      -LEFT]";
2820  input a$
2830  a1$=left$(a$,1)
2840  a2$=right$(a$,1)
2850  if len(a$) <> 2 then goto 2870
2860  if a1$>chr$(64) and a1$<chr$(75) a
      nd a2$>chr$(48) and a2$<chr$(56)go
      to2910
2870  print"[HOME][23xCRSR-DOWN]";
2880  print "dit[SPACE]is[SPACE]een[SPAC
      E]niet[SPACE]bestaand[SPACE]vak."
2890  for x = 1 to 1000: next
2900  goto 2800
2910  a1=(asc(a1$)-64)
2920  a2=(asc(a2$)-48)
2930  if b$(a2,a1) <> "r" then goto 2980
2940  print"[HOME][23xCRSR-DOWN]";
2950  print"hier[SPACE]is[SPACE]al[SPACE
      ]eens[SPACE]op[SPACE]geschoten."
2960  for x = 1 to 1000: next x
2970  goto 2800
2980  if b$(a2,a1) = "vl" then c=1
2990  if b$(a2,a1) = "kr" then c=2
3000  if b$(a2,a1) = "oz" then c=3
3010  if b$(a2,a1) = "mv" then c=4
3020  b=b+1
3030  print"[HOME]";
3040  for x= 1to a2*3-1
3050  print"[CRSR-DOWN]";
3060  next x
3070  for x=1to a1*3-1
3080  print "[CRSR-RIGHT]";
3090  next x
3100  b$(a2,a1) = "r"
3110  if c=1 then goto 3170
3120  if c=2 then goto 3200
3130  if c=3 then goto 3230
3140  if c=4 then goto 3260
3150  print "[COM-B]"
3160  goto 2800
3170  c=0
```

```
3180  vl = vl+1
3190  goto 3280
3200  c=0
3210  kr = kr+1
3220  goto 3280
3230  c=0
3240  oz = oz+1
3250  goto 3280
3260  c=0
3270  mv = mv+1
3280  rs=rs+1
3290  print "[CTRL-1]";"[COM-+]";"[CTRL-
      6]"
3300  if vl=5 then poke v+39,0
3310  if kr=4 then poke v+40,0
3320  if oz=3 then poke v+41,0
3330  if mv=2 then poke v+42,0
3340  if rs=14 then goto 3360
3350  goto 2800
3360  print"[HOME][22xCRSR-DOWN]"
3370  print"[23xSPACE]"
3380  print "[CRSR-UP]";
3390  print "gezonken[SPACE]in";b;"schot
      en."
3400  input "nog[SPACE]een[SPACE]spellet
      je[SPACE](j[SPACE]of[SPACE]n)";a$
3410  if a$ = "n" then goto 4070
3420  if a$ <> "n" and a$ <> "j" then go
      to 3500
3430  poke v+21,0: print "[SHIFT-CLR]"
3440  poke v+39,5  :rem kleur sprite 0
3450  poke v+40,5  :rem kleur sprite 1
3460  poke v+41,5  :rem kleur sprite 2
3470  poke v+42,5  :rem kleur sprite 3
3480  vl=0:kr=0:oz=0:mv=0:rs=0:b=0
3490  goto 1330
3500  poke v+21,0
3510  print "[SHIFT-CLR]"
3520  print "het[SPACE]antwoord[SPACE]mo
      et[SPACE][CTRL-9]j[CTRL-0][SPACE]o
      f[SPACE][CTRL-9]n[CTRL-0][SPACE]zi
      jn"
3530  print "[2xCRSR-DOWN]";
3540  goto 3400
3550  rem    instellen sprite gegevens
3560  rem    ========================
3570  mem=192 : smem=64*mem : v=53248
3580  rem      inlezen sprite data
3590  rem      ==================
3600  for n = 0to(64*4)-1
3610  check = ((n+1)/64)-int((n+1)/64)
3620  if check = 0then goto 3650
3630  read d
3640  poke smem+n,d
3650  next n
3660  poke 2040,192:rem sprite 0pointer
3670  poke 2041,193:rem sprite 1 pointer
3680  poke 2042,194:rem sprite 2 pointer
3690  poke 2043,195:rem sprite 3 pointer
3700  poke 2044,194:rem sprite 4 pointer
3710  poke v+39,5  :rem kleur sprite 0
3720  poke v+40,5  :rem kleur sprite 1
3730  poke v+41,5  :rem kleur sprite 2
3740  poke v+42,5  :rem kleur sprite 3
3750  poke v+43,0 :rem kleur sprite 4
3760  poke v+23,31 :rem hoogte bepaling
3770  poke v+29,31 :rem breedte bepaling
3780  return
3790  rem      data vliegdekmoederschip
3800  rem      =======================
3810  data 0,0,0,0,0,0,0,0,0,0,0,0,0,0
3820  data 0,0,0,0,0,0,0,0,0,16,0,0,12
      4,0
3830  data 0,16,0,0,56,0,0,60,0,28,127,3
      2,62,127,240
3840  data 62,63,0,62,18,0,255,255,255,2
      55,255,255,31,255,240
3850  data 31,255,224
3860  rem      data kruiser
3870  rem      ===========
3880  data 0,0,0,0,0,0,0,0,0,0,0,0,0,0
3890  data 0,0,0,0,0,0,0,0,0,0,0,0,0,0
3900  data 0,0,0,0,0,0,0,0,0,32,0,24,1
      12,0
3910  data 248,248,56,24,248,63,24,248,5
      6,255,255,255,255,255,254
3920  data 127,255,252
3930  rem      data onderzeeboot
3940  rem      ================
3950  data 0,0,0,0,0,0,0,0,0,0,0,0,0,0
3960  data 0,0,0,0,0,0,0,0,0,0,0,0,0,0
3970  data 0,0,0,0,0,0,0,0,0,0,0,0,0,0
3980  data 0,0,0,0,192,0,0,192,0,0,192,0
      ,31,255,254
3990  data 127,255,255
4000  rem      data mijnenveger
4010  rem      ==============
4020  data 0,0,0,0,0,0,0,0,0,0,0,0,0,0
4030  data 0,0,0,0,0,0,0,0,0,0,0,0,0,0
4040  data 3,224,0,0,128,0,0,128,0,1,192
      ,0,1,192,0
4050  data 1,240,0,1,240,0,1,240,0,255,2
      55,255,127,255,254
4060  data 63,255,252
4070  poke v+21,0
4080  print "[SHIFT-CLR]"
4090  end
```

** EINDE LISTING zeeslag **



Checksum Zeeslag

regel 100	5
regel 110	55
regel 120	66
regel 130	55
regel 140	5
regel 150	143
regel 160	20
regel 170	161
regel 180	194
regel 190	143
regel 200	159
regel 210	143
regel 220	143
regel 230	86
regel 240	213
regel 250	158
regel 260	143
regel 270	143
regel 280	5
regel 290	143
regel 300	143
regel 310	95
regel 320	112
regel 330	188
regel 340	194
regel 350	185
regel 360	13
regel 370	47
regel 380	179
regel 390	185
regel 400	101
regel 410	251
regel 420	242
regel 430	109
regel 440	90
regel 450	180
regel 460	225
regel 470	248
regel 480	47
regel 490	58
regel 500	63
regel 510	58
regel 520	35
regel 530	244
regel 540	46
regel 550	171
regel 560	22
regel 570	56
regel 580	43
regel 590	63
regel 600	58
regel 610	12
regel 620	61
regel 630	58
regel 640	179
regel 650	60
regel 660	58
regel 670	30
regel 680	59
regel 690	58
regel 700	200
regel 710	58
regel 720	90
regel 730	207
regel 740	210
regel 750	213
regel 760	216
regel 770	164
regel 780	209
regel 790	215
regel 800	222
regel 810	34
regel 820	58
regel 830	8
regel 840	86
regel 850	112
regel 860	119
regel 870	11
regel 880	153
regel 890	133
regel 900	27
regel 910	118
regel 920	127
regel 930	153
regel 940	114
regel 950	116
regel 960	58
regel 970	98
regel 980	195
regel 990	187
regel 1000	58
regel 1010	76
regel 1020	181
regel 1030	117
regel 1040	58
regel 1050	8
regel 1060	171
regel 1070	25
regel 1080	153
regel 1090	255
regel 1100	185
regel 1110	58
regel 1120	193
regel 1130	58
regel 1140	176
regel 1150	66
regel 1160	131
regel 1170	171
regel 1180	170
regel 1190	240
regel 1200	217
regel 1210	176
regel 1220	30
regel 1230	16
regel 1240	29
regel 1250	255
regel 1260	8
regel 1270	86
regel 1280	112
regel 1290	135
regel 1300	82
regel 1310	153
regel 1320	132
regel 1330	168
regel 1340	150
regel 1350	193
regel 1360	64
regel 1370	219
regel 1380	218
regel 1390	248
regel 1400	181
regel 1410	122
regel 1420	3
regel 1430	51
regel 1440	94
regel 1450	149
regel 1460	87
regel 1470	209
regel 1480	130
regel 1490	85
regel 1500	86
regel 1510	59
regel 1520	149
regel 1530	101
regel 1540	209
regel 1550	218
regel 1560	58
regel 1570	156
regel 1580	122
regel 1590	7
regel 1600	51
regel 1610	92
regel 1620	41
regel 1630	229
regel 1640	218
regel 1650	231
regel 1660	179
regel 1670	148
regel 1680	87
regel 1690	204
regel 1700	218
regel 1710	85
regel 1720	86
regel 1730	57
regel 1740	55
regel 1750	225
regel 1760	218
regel 1770	199
regel 1780	182
regel 1790	148
regel 1800	101
regel 1810	204
regel 1820	218
regel 1830	119
regel 1840	144
regel 1850	122
regel 1860	16
regel 1870	51
regel 1880	90
regel 1890	40
regel 1900	238
regel 1910	218
regel 1920	231
regel 1930	188
regel 1940	147
regel 1950	87
regel 1960	216
regel 1970	218
regel 1980	76
regel 1990	86
regel 2000	55
regel 2010	54
regel 2020	234
regel 2030	218
regel 2040	190
regel 2050	191
regel 2060	147
regel 2070	101
regel 2080	216
regel 2090	218
regel 2100	33
regel 2110	83
regel 2120	122
regel 2130	7
regel 2140	51
regel 2150	88
regel 2160	39
regel 2170	229
regel 2180	218
regel 2190	222
regel 2200	179
regel 2210	146
regel 2220	87
regel 2230	210
regel 2240	218
regel 2250	85
regel 2260	86
regel 2270	53
regel 2280	53
regel 2290	225
regel 2300	218
regel 2310	190
regel 2320	182
regel 2330	146
regel 2340	101
regel 2350	210
regel 2360	218
regel 2370	192
regel 2380	107
regel 2390	72
regel 2400	86
regel 2410	171
regel 2420	148
regel 2430	112
regel 2440	152
regel 2450	24
regel 2460	153
regel 2470	85
regel 2480	218
regel 2490	51
regel 2500	61
regel 2510	166
regel 2520	240
regel 2530	218
regel 2540	6
regel 2550	24
regel 2560	166
regel 2570	240
regel 2580	218
regel 2590	6
regel 2600	47
regel 2610	219
regel 2620	24
regel 2630	195
regel 2640	207
regel 2650	218
regel 2660	1
regel 2670	54
regel 2680	144
regel 2690	158
regel 2700	161
regel 2710	164
regel 2720	167
regel 2730	162
regel 2740	209
regel 2750	217
regel 2760	216
regel 2770	34
regel 2780	171
regel 2790	107
regel 2800	178
regel 2810	157
regel 2820	234
regel 2830	35
regel 2840	37
regel 2850	155
regel 2860	28
regel 2870	178
regel 2880	180
regel 2890	221
regel 2900	83
regel 2910	55
regel 2920	59
regel 2930	80
regel 2940	178
regel 2950	198
regel 2960	53
regel 2970	83
regel 2980	184
regel 2990	180
regel 3000	193
regel 3010	188
regel 3020	17
regel 3030	43
regel 3040	142
regel 3050	41
regel 3060	218
regel 3070	141
regel 3080	53
regel 3090	218
regel 3100	16
regel 3110	172
regel 3120	167
regel 3130	171
regel 3140	175
regel 3150	156
regel 3160	83
regel 3170	37
regel 3180	209
regel 3190	86
regel 3200	37
regel 3210	199
regel 3220	86
regel 3230	37
regel 3240	223
regel 3250	86
regel 3260	37
regel 3270	211
regel 3280	215
regel 3290	47
regel 3300	26
regel 3310	12
regel 3320	24
regel 3330	18
regel 3340	67
regel 3350	83
regel 3360	102
regel 3370	221
regel 3380	169
regel 3390	19
regel 3400	75
regel 3410	47
regel 3420	228
regel 3430	0
regel 3440	183
regel 3450	176
regel 3460	178
regel 3470	180
regel 3480	224
regel 3490	80
regel 3500	86
regel 3510	112
regel 3520	17
regel 3530	58
regel 3540	80
regel 3550	104
regel 3560	132
regel 3570	136
regel 3580	149
regel 3590	22
regel 3600	204
regel 3610	52
regel 3620	201
regel 3630	203
regel 3640	49
regel 3650	208
regel 3660	22
regel 3670	25
regel 3680	28
regel 3690	31
regel 3700	32
regel 3710	183
regel 3720	176
regel 3730	178
regel 3740	180
regel 3750	177
regel 3760	93
regel 3770	152
regel 3780	142
regel 3790	39
regel 3800	71
regel 3810	187
regel 3820	89
regel 3830	68
regel 3840	43
regel 3850	115
regel 3860	206
regel 3870	107
regel 3880	187
regel 3890	187
regel 3900	138
regel 3910	21
regel 3920	170
regel 3930	57
regel 3940	156
regel 3950	187
regel 3960	187
regel 3970	187
regel 3980	10
regel 3990	173
regel 4000	227
regel 4010	95
regel 4020	187
regel 4030	187
regel 4040	214
regel 4050	117
regel 4060	121
regel 4070	86
regel 4080	112
regel 4090	128

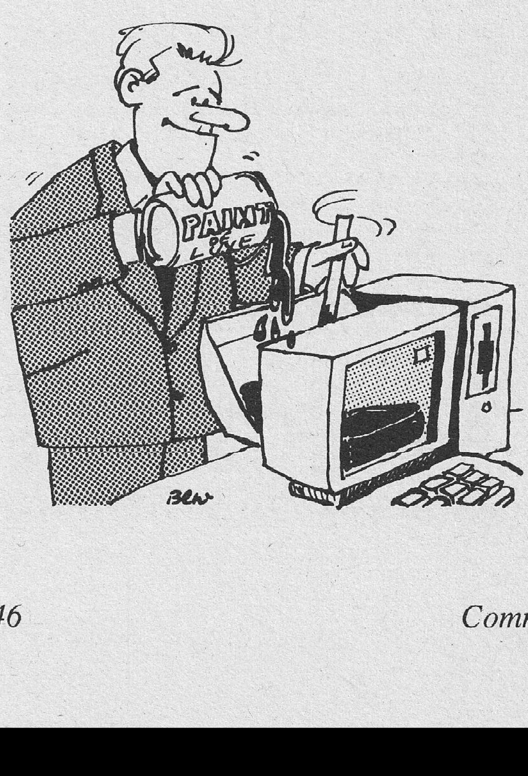
PRINT OUT C-128 met Character Editor

Character-editor 128

We hebben in de afgelopen jaren al verscheidene malen soortgelijke programma's geplaatst. Er blijkt echter altijd weer veel vraag naar te zijn. Computergebruikers blijken mensen te zijn met veel fantasie en geduld. Voor deze mensen is dit programma uitermate geschikt. U kunt U eigen karakters maken, en deze later gaan gebruiken. Het programma, van Xander Bommele uit Delft, sprekkt verder voor zich.

```
10    rem gemaakt door :arnout kuiper
20    rem
30    rem              oostvoorne
40    rem
100   poke56,127:clr
110   ifpeek(49157)<>192thengosub1450
120   poke53280,0:poke53281,0:vi=1225:poke650,128:poke808,251
130   fori=0to63:poke704+i,0:next:poke53271,0:poke53277,0
140   poke2040,11:poke53269,1:poke53248,120:poke53249,91:poke53287,6:poke53276,0
150   ba=0:ka=32:ta$="[19xCRSR-RIGHT]"
160   me$="[HOME][20xCRSR-DOWN][CRSR-RIGHT]"
170   cm$=me$+"[38xSPACE]"
180   printchr$(8);chr$(142);"[CTRL-7][SHIFT CLR][CTRL-8]UCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCI";
190   print"B[CTRL-3][8xSPACE]character-editor[SPACE]deluxe[7xSPACE][CTRL-8]B";
200   print"B[CTRL-3][5xSPACE](c)[SPACE]1988[SPACE]by[SPACE]arnout's[SPACE]software[4xSPACE][CTRL-8]B";
210   print"JCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCK"
220   printme$:print"[2xCRSR-UP]UCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCI";
230   print"B[38xSPACE]B";
240   print"JCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCK";
250   print"[HOME][5xCRSR-DOWN]";ta$;"[CTRL 7]bank[6xSPACE]:[CTRL-8]";right$("[3xSPACE]"+str$(ba),3)
260   printta$;"[CTRL-7]character[SPACE]:[CTRL-8]";right$("[3xSPACE]"+str$(ka),3)
270   sys49152,ba,ka,vi
280   x=0:y=0:printcm$;
290   pokevi+x+y*40+54272,1
300   geta$:ifa$=""then300
310   pokevi+x+y*40+54272,6
320   ifa$="[CRSR-DOWN]"theny=(y+1)and7
330   ifa$="[CRSR-UP]"theny=(y-1)and7
340   ifa$="[CRSR-RIGHT]"thenx=(x+1)and7
350   ifa$="[CRSR-LEFT]"thenx=(x-1)and7
360   ifa$="[SPACE]"thensys49158,x,y,vi
370   ifa$=chr$(133)thensys49167,4,vi
380   ifa$=chr$(137)thensys49167,3,vi
390   ifa$=chr$(134)thensys49167,1,vi
400   ifa$=chr$(138)thensys49167,2,vi
410   ifa$="i"thensys49161,vi
420   ifa$="g"then560
430   ifa$="b"then620
440   ifa$="p"then650
450   ifa$="c"then850
460   ifa$="n"thensys49155:goto270
470   ifa$="s"then760
480   ifa$="l"then800
490   ifa$="[SHIFT-CLR]"thensys49173,vi
500   ifa$=chr$(3)thenprintme$;"[16xSPACE]the[SPACE]end!";:sys64738
510   ifa$=chr$(141)thenk=ka:b=ba:goto750
520   ifa$="_"then610
530   ifa$="x"thensys49176,1,vi
540   ifa$="y"thensys49176,0,vi
550   goto290
560   printme$;"get[SPACE]character[SPACE]:[SPACE]key[SPACE](1)[SPACE]or[SPACE]code[SPACE](2)";
570   geta$:ifa$<>"1"anda$<>"2"anda$<>chr$(136)then570
580   printcm$;me$;:ifa$=chr$(136)then290
590   ifa$="1"then610
600   print"code[SPACE]:[SPACE]";:gosub890:goto250
610   printme$;"key[SPACE]:[SPACE]";:gosub1000:goto250
620   printme$;"bank[SPACE]:";
630   geta$:ifa$<"0"ora$>"3"then630
640   ba=val(a$):goto250
650   printme$;"at[SPACE]same[SPACE]location?[SPACE](y/n)";
660   geta$:ifa$<>"y"anda$<>"n"anda$<>chr$(136)then660
670   printcm$;:k=ka:b=ba:ifa$=chr$(136)then290
680   ifa$="y"then750
690   printme$;"put[SPACE]character[SPACE]:[SPACE]key[SPACE](1)[SPACE]or[SPACE]code(2)";
700   geta$:ifa$<>"1"anda$<>"2"anda$<>chr$(136)then700
710   printcm$;me$;:ifa$=chr$(136)then290
720   ifa$="1"then740
730   print"code[SPACE]:[SPACE]";:gosub890:goto750
740   print"key[SPACE]:[SPACE]";:gosub1000
750   sys49164,ba,ka:ka=k:ba=b:goto250
760   printme$;"save,[SPACE]filename[SPACE]:[SPACE]";:gosub1120:ifb$=""then290
770   open15,8,15
780   open1,8,2,b$+",p,w":input#15,en,a$,c$,d$:ifen<20thensys49170:goto840
790   goto830
800   printme$;"load,[SPACE]filename[SPACE]:[SPACE]";:gosub1120:ifb$=""then290
810   open15,8,15
820   open1,8,2,b$+",p,r":input#15,en,a$,c$,d$:ifen<20thensys49179:goto840
830   printme$;cm$;me$;en;",";a$;",";c$;",";d$;"[HOME]":poke198,0:wait198,1
840   close1:close15:poke53280,0:printcm
```

```
     $;:goto250
850  printme$;"copy,[SPACE]bank[SPACE]:";
860  geta$:ifa$<"0"ora$>"4"anda$<>chr$(136)then860
870  ifa$=chr$(136)then280
880  k=ka:b=ba:ba=val(a$):goto690
890  b$=""
900  geta$:ifa$<>chr$(13)anda$<>chr$(20)anda$<"0"ora$>"9"then900
910  ifa$=chr$(13)andb$<>""then970
920  ifa$=chr$(20)andb$<>""thenb$=left$(b$,len(b$)-1):goto960
930  ifa$=chr$(20)then900
940  iflen(b$)<3thenb$=b$+a$:printa$;:goto900
950  goto900
960  printleft$("[3xCRSR-LEFT]",len(b$)+1);b$;"[2xSPACE][2xCRSR-LEFT]";:goto900
970  ka=val(b$)
980  ifka>127thenka=ka-128:goto980
990  return
1000 geta$:ifa$=""then1000
1010 a=asc(a$)
1020 ifa>63anda<96thena=a-64:goto1110
1030 ifa>31anda<64then1110
1040 ifa>95anda<128thena=a-32:goto1110
1050 ifa>191anda<224thena=a-128:goto1110
1060 ifa>159anda<192thena=a-64:goto1110
1070 ifa>223anda<255thena=a-128:goto1110
1080 ifa=255thena=94:goto1110
1090 ifa>-1anda<32thenba=(baand2)+1:goto1110
1100 ifa>127anda<160thenba=(baand2)+1:a=a-64:goto1110
1110 ka=a:return
1120 b$=""
1130 geta$:ifa$<>chr$(13)anda$<>chr$(20)anda$<>"[SPACE]"anda$<"#"ora$>"z"then1130
1140 ifa$=chr$(13)thenprintme$;cm$;"[HOME]":return
1150 ifa$=chr$(20)andb$<>""thenb$=left$(b$,len(b$)-1):goto1190
1160 ifa$=chr$(20)then1130
1170 iflen(b$)<16thenb$=b$+a$:printa$;:goto1130
1180 goto1130
1190 printleft$("[16xCRSR-LEFT]",len(b$)+1);b$;"[2xSPACE][2xCRSR-LEFT]";:goto1130
1200 data 49152, 23
1210 data"4c1ec04c79c04cadc04cc7c04cd9c04ce7c04c5ac14c97c14ca6c14c"
1220 data"37c22002c2b1fb9900cfc8c008d0f620fdae209ead20f7b7a200a000"
1230 data"bd00cf3971c0f005a9a04c47c0a92e9114c8c008d0eaa51418692885"
1240 data"149002e615e8e008d0d8a200a000b900cf9dc002e8e8e8c8c008d0f2"
1250 data"60804020100804020178a9008d0edca9338501a9d085fca98085fea0"
1260 data"0084fb84fdb1fb91fdc8d0f9e6fce6fea5fcc9e0d0efa9378501a901"
1270 data"8d0edc58602031c286fb2031c286fca4fca6fbb900cf5d71c09900cf"
1280 data"4c2bc0a000b900cf49ff9900cfc8c008d0f34c2bc02002c2b900cf91"
1290 data"fbc8c008d0f6602031c2e001d0034c42c1e002d0034c2ac1e003d003"
1300 data"4c17c1a200187e00cf9008bd00cf697f9d00cfe8e008d0ed4c2bc0a2"
1310 data"00183e00cf9003fe00cfe8e008d0f24c2bc0ad00cf48a001b900cf99"
1320 data"ffcec8c008d0f5688d07cf4c2bc0ad07cf48a006b900cf9901cf88c0"
1330 data"ffd0f5688d00cf4c2bc0a90b8d11d0a20120c9ffa90020d2ffa93020"
1340 data"d2ffa98085fca00084fbee20d0b1fb20d2ffc8d0f5e6fca5fcc990d0"
1350 data"ed20ccffa90120c3ffa91b8d11d060a000a9009900cfc8c008d0f64c"
1360 data"2bc02031c2e001f020a000a207b900cf9d10cfcac8c008d0f4a200bd"
1370 data"10cf9d00cfe8e008d0f54c2bc0a000b900cf85fba90085fda200a5fb"
1380 data"3d71c0f007a5fd1dfac185fde8e008d0eda5fd9900cfc8c008d0d84c"
1390 data"2bc00102040810204080 2031c286fc2031c286fba90085fe06fc06fc"
1400 data"18a5fc698085fc06fb26fe06fb26fe06fb26fea5fe1865fc85fca000"
1410 data"6020fdae4c9eb7a90b8d11d0a20120c6ff20e4ff20e4ffa98085fca0"
1420 data"0084fbee20d020e4ff91fbc8d0f5e6fca5fcc990d0ed20ccffa90120"
1430 data"c3ffa91b8d11d06000ce20c506ec00c120e700e41640000000abc0ff"
1440 data 87343
1450 reads,a
1460 fori=1toa:reada$:forj=0to27:m$=mid$(a$,j*2+1,2)
1470 l=asc(left$(m$,1))-48:r=asc(right$(m$,1))-48
1480 g=16*(l+7*(l>9))+r+7*(r>9):t=t+g:pokes+j,g:next:s=s+28:next
1490 reada:ift<>athenprint"fout[SPACE]in[SPACE]data":end
1500 return
```



** EINDE LISTING character-editor

PRINT OUT C-16 met Boekhouden

Checksum c16

```
10     rem ******************************
20     rem syntax.checksum
30     rem voor c-16 & plus/4
40     rem
50     rem syntax testen met 'sys 1536'
60     rem
70     rem v.851128.16        jan bodzinga
80     rem ******************************
90     i=1536        :rem beginadres
100    reada:ifa>=0then pokei,a:i=i+1:got
       o100
110    print"data[SPACE]is[SPACE]weggezet"
120    print"cheksum[SPACE]printen[SPACE]
       met[SPACE]'sys[SPACE]1536'
130    end
200    data 165, 43,166, 44,133
210    data  31,134, 32,169,147
220    data  32,210,255,160,  0
230    data 240,  3, 32, 73,  6
240    data  32, 73,  6,208,  1
250    data  96, 72,152, 32,131
260    data   6,168,104,234, 32
270    data  81,  6, 32, 73,  6
280    data 240, 12,201, 32,240
290    data 247, 24,101,252,133
300    data 252, 76, 37,  6,166
310    data 252,169,  0,132,253
320    data  32, 95,164,169, 13
330    data  32,210,255,164,253
340    data  76, 17,  6,200,208
350    data   2,230, 32,177, 31
360    data  96,162,  0,189,123
370    data   6,240,  6, 32,210
380    data 255,232,208,245, 32
390    data  73,  6,170, 32, 73
400    data   6,132,253, 32, 95
410    data 164,162,  3,169, 32
420    data  32,210,255,202,208
430    data 250,169,  0,133,252
440    data 164,253, 96, 82, 69
450    data  71, 69, 76, 32,  0
460    data   0, 72,138, 72, 32
470    data 225,255,240,251,104
480    data 170,104, 96, -1
```

** EINDE LISTING checks16

Checksum Checksum C-16

REGEL 10	249	REGEL 290	248
REGEL 20	247	REGEL 300	118
REGEL 30	121	REGEL 310	204
REGEL 40	143	REGEL 320	165
REGEL 50	75	REGEL 330	252
REGEL 60	143	REGEL 340	106
REGEL 70	8	REGEL 350	98
REGEL 80	249	REGEL 360	163
REGEL 90	103	REGEL 370	45
REGEL 100	2	REGEL 380	0
REGEL 110	245	REGEL 390	58
REGEL 120	237	REGEL 400	108
REGEL 130	128	REGEL 410	159
REGEL 200	210	REGEL 420	245
REGEL 210	208	REGEL 430	202
REGEL 220	142	REGEL 440	176
REGEL 230	1	REGEL 450	12
REGEL 240	3	REGEL 460	54
REGEL 250	157	REGEL 470	43
REGEL 260	155	REGEL 480	1
REGEL 270	215		
REGEL 280	186		

Boekhouden

Een eenvoudig boekhoudprogramma voor het gezin,en één die op een eenvoudige manier is te gebruiken. Het programma spreekt voor zich zelf. Eventuele aanpassingen zijn eenvoudig aan te brengen. De bedenker van dit programma is Chr.v.Bruggen uit Groningen.

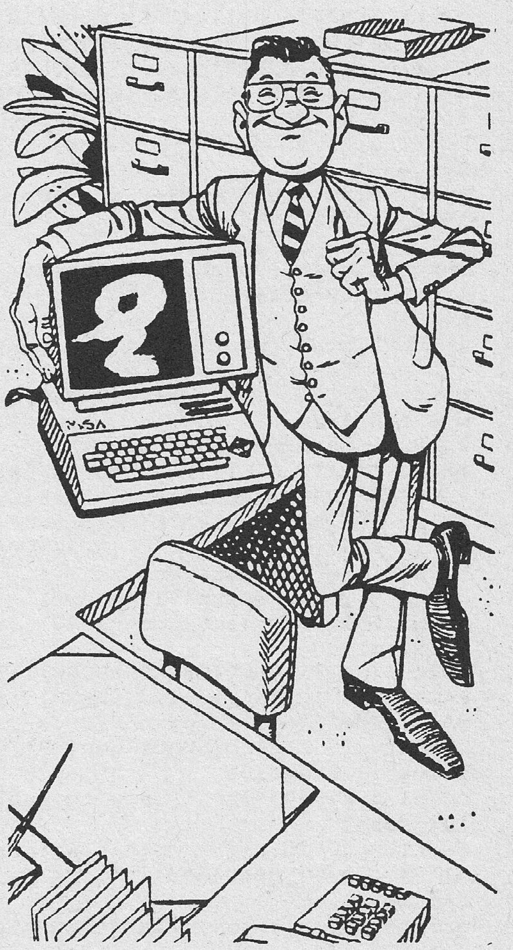
```
10     rem boekhoudprogramma voor c16 doo
       r  chris van bruggen uit groningen
20     iffre(x)>12277thendi=1200:elsedi=1
       50
30     print"[SHIFT-CLR]";tab(10)"[3xCRSR
       -DOWN]boekhoudprogramma":printtab(
       10)"[2xCRSR-DOWN]voor[SPACE]";di;"
       [SPACE]posten"
40     printtab(10)"[3xCRSR-DOWN]even[SPA
       CE]geduld[SPACE]a.u.b."
50     dimc(di):dimb$(di):dima$(14):dime(
       di):fortf=1to14:reada$(tf):nexttf
60     y=1
70     rem openingsscherm
80     print"[SHIFT-CLR][2xCRSR-DOWN]invo
       er[26xSPACE]toets[SPACE]i"
90     print"[CRSR-DOWN]oude[SPACE]data[S
       PACE]laden[SPACE]van[SPACE]cassett
       e:[3xSPACE]toets[SPACE]c"
100    print"[CRSR-DOWN]oude[SPACE]data[S
       PACE]laden[SPACE]van[SPACE]disk:[7
       xSPACE]toets[SPACE]d"
110    geta$:ifa$="i"then130
120    ifa$="c"ora$="d"then1410:else110
130    poke2025,128:print"[SHIFT-CLR]":x=
       y:gosub160:gosub230:gosub350:gosub
       620:gosub630:gosub390
140    y=x:goto660
150    rem schermen bedieningvoorschrift
160    poke2021,24:poke2022,24:poke2023,0
170    print"[SHIFT-CLR][CTRL-9]korrektie
       [6xSPACE]=[SPACE][3xSPACE]uitvoer
       [8xSPACE]=[SPACE]*[SPACE][CTRL-0]"
       ;:return
180    poke2021,24:poke2022,24:poke2023,0
190    print"[SHIFT-CLR][CTRL-9]korrektie
       [6xSPACE]=[SPACE][22xSPACE]";:ret
       urn
200    poke2021,24:poke2022,24:poke2023,0
210    print"[SHIFT-CLR][CTRL-9]onmiddell
       ijk[SPACE]stoppen[SPACE]=[SPACE]@[
       16xSPACE]";:return
220    rem scherm keuze rubrieken
230    poke2022,12:poke2021,23:poke2023,0
240    print"[SHIFT-CLR][40xCOM-I]"
250    printtab(16)"[CTRL-9]inkomsten":pr
       inta$(1);tab(17)"=[SPACE]1B";a$(2)
       ;tab(36)"=[SPACE]2"
260    print"CCCCCCCCCCCCCCCCCCC[SHIFT-+
       ]CCCCCCCCCCCCCCCCCCC"
270    printtab(16)"[CTRL-9]uitgaven[SPAC
       E][CTRL-0]":printa$(3);tab(17)"=[S
       PACE]3B";a$(4);tab(36)"=[SPACE]4"
280    printa$(5);tab(17)"=[SPACE]5B";a$(
       6);tab(36)"=[SPACE]6"
290    printa$(7);tab(17)"=[SPACE]7B";a$(
       8);tab(36)"=[SPACE]8"
300    printa$(9);tab(17)"=[SPACE]9B";a$(
       10);tab(36)"=10"
```

```
310   printa$(11);tab(17)"=11B";a$(12);t
      ab(36)"=12"
320   printa$(13);tab(17)"=13B";a$(14);t
      ab(36)"=14"
330   printtab(20)"B":return
340   rem scherm benaming posten
350   poke2022,0:poke2021,3:poke2023,0
360   print"[SHIFT-CLR]onder[SPACE]rubri
      ek:":print"onder[SPACE]maand:":pri
      nt"bedrag:":print"omschrijving:"
370   return
380   rem invoer gegevens
390   poke2021,3:poke2022,0:poke2023,16:
      print"[SHIFT-CLR]";
400   gosub540:ifa$="*"ands=0thenreturn
410   ifa<1ora>14thenprint"[SHIFT-CLR]";
      :goto400
420   c$=a$:iflen(c$)=1thenc$="0"+c$
430   gosub540:ifa$=""thenprint"[SHIFT-
      CLR]";:goto400
440   ifa<1ora>12thenprint"[CRSR-UP]";ch
      r$(27);"d";:goto430
450   d$=a$:iflen(d$)=1thend$="0"+d$
460   gosub540:ifa$=""thenprint"[SHIFT-
      CLR]";:goto400
470   c(x)=val(a$):ifc(x)<0.01orc(x)>999
      9.99thenprint"[CRSR-UP]";chr$(27);
      "d";:goto460
480   gosub540:ifa$=""thenprint"[SHIFT-
      CLR]";:goto400
490   iflen(a$)>20thena$=left$(a$,20)
500   b$(x)=c$+"[SPACE]"+d$+"[SPACE]"+a$
510   ifdi-x<2orfre(x)<50thenprint"gehe
      ugen[SPACE]is[SPACE]vol":print"n
      a[SPACE]volgende[SPACE]post":else
      520
520   ifs=1thengosub600:gosub620:gosub63
      0:goto570
530   gosub600:gosub620:gosub630:gosub55
      0:goto390
540   printtab(18);:a$="":inputa$:a=val(
      a$):return
550   geta$:ifa$="j"thenx=x+1:gosub600:p
      rint"[SHIFT-CLR]":return
560   ifa$="n"thengosub600:print"[SHIFT-
      CLR]":gosub620:gosub590:return:els
      e550
570   geta$:ifa$="j"thenreturn
580   ifa$="n"thengosub600:print"[SHIFT-
      CLR]":gosub620:gosub590:goto390:el
      se570
590   c(x)=0:b$(x)="":print"[CRSR-DOWN]"
      ;chr$(27);"d":return
600   poke2021,4:poke2023,0:poke2022,4:p
      rint"[SHIFT-CLR]invoer[SPACE]akkoo
      rd[4xSPACE]?[SPACE](j)a[SPACE]of[S
      PACE](n)ee":return
610   rem tijdelijke uitvoer
620   poke2021,11:poke2022,5:poke2023,0:
      print"[HOME]";:return
630   printchr$(27);"w":print"[CTRL-9]";
      x;"[CTRL-0]";tab(6)b$(x);tab(32);
635 ifc(x)=0thenprint"voer[SPACE]in":r
      eturn
640   printusing"####.##";c(x);:return
650   rem uitvoer
660   poke2025,128:poke2021,24:poke2022,
      0:poke2023,0
670   print"[SHIFT-CLR][2xCRSR-DOWN]uitv
      oer[SPACE]gesorteerd[14xSPACE]=[SP
      ACE]a"
680   print"[CRSR-DOWN]wijziging[SPACE]o
      f[SPACE]weghalen[SPACE]posten[4xSP
      ACE]=[SPACE]b"
690   print"[CRSR-DOWN]opslaan[SPACE]op[
      SPACE]tape[17xSPACE]=[SPACE]c"
700   print"[CRSR-DOWN]opslaan[SPACE]op[
      SPACE]disk[17xSPACE]=[SPACE]d"
710   print"[CRSR-DOWN]opnieuw[SPACE]inv
      oeren[16xSPACE]=[SPACE]e"
720   geta$:ifa$="a"then840
730   ifa$="b"then760
740   ifa$="c"ora$="d"then1530
750   ifa$="e"then130:else720
760   print"[SHIFT-CLR]welk[SPACE]rekeni
      ng[SPACE]nummer[SPACE]";:inputa$:z
      =val(a$):ifz<1then760
770   ifz>x-1thenprint"[2xCRSR-DOWN]reke
      ningnummer[SPACE]is[SPACE]niet[SPA
      CE]aanwezig":fort=1to1000:next:got
      o660
780   x=z:gosub620:gosub630:gosub790:x=y
      :goto660
790   print:print"[2xCRSR-DOWN](v)erwijd
      eren[SPACE](w)ijzigen[SPACE]of[SPA
      CE](a)fbreken[SPACE]?";
800   geta$:ifa$="v"thenb$(x)="leeg":c(x
      )=0:print"[CRSR-DOWN][CTRL-9]";x;"
      [CTRL-0]";b$(x):fort=1to1000:next:
      return
810   ifa$="w"thenprintchr$(27);"d":s=1:
      gosub180:gosub230:gosub350:gosub39
      0:s=0:return
820   ifa$="a"thenreturn:else800
830   rem sorteerfunktie
840   x=0:m=1:u=0:v=0:print"[SHIFT-CLR][
      2xCRSR-DOWN]uitvoer":print"[2xCRSR
      -DOWN]sorteren[SPACE]enkele[SPACE]
      rubriek[13xSPACE]=[SPACE]a"
850   print"[CRSR-DOWN]sorteren[SPACE]en
      kele[SPACE]maand[15xSPACE]=[SPACE]
      b"
860   print"[CRSR-DOWN]sorteren[SPACE]en
      kele[SPACE]rubriek[SPACE]en[SPACE]
      maand[4xSPACE]=[SPACE]c"
870   print"[CRSR-DOWN]sorteren[SPACE]al
      le[SPACE]posten[16xSPACE]=[SPACE]d
      "
880   print"[CRSR-DOWN]ongesorteerde[SPA
      CE]uitvoer[15xSPACE]=[SPACE]e"
890   print"[CRSR-DOWN]nieuwe[SPACE]keus
      [25xSPACE]=[SPACE]f"
900   gete$:ife$="a"then:gosub230:gosub6
      20:gosub960:gosub200:gosub980:goto
      1180
910   ife$="b"thengosub970:gosub200:gosu
      b980:goto1180
920   ife$="c"thengosub230:gosub620:gosu
      b960:gosub970:gosub200:gosub980:go
      to1180
930   ife$="d"thengosub200:d=1:e=0:gosub
      980:goto1180
940   ife$="e"then:gosub200:gosub980:got
      o1180
950   ife$="f"thenx=y:goto660:else900
960   print"[SHIFT-CLR][2xCRSR-DOWN]kies
```

```
     [SPACE]een[SPACE]rubriek[SPACE]";:
     inputa$:d=val(a$):ifd<1ord>14then9
     60:elsereturn
970  print"[SHIFT-CLR][2xCRSR-DOWN]kies
     [SPACE]een[SPACE]maand[SHIFT-SPACE
     ]";:inputa$:e=val(a$):ife<1ore>12t
     hen970:elsereturn
980  poke2022,0:poke2021,23:print"[SHIF
     T CLR]":poke2025,0:h=1
990  x=x+1:ifx=ythen1110
1000 f=val(left$(b$(x),2)):g=val(mid$(b
     $(x),4,2))
1010 ifb$(x)="leeg"ande$<>"e"thenf=1:g=
     0
1020 getb$:ifb$="@"then1130
1030 ife$="e"then1070
1040 ife$="a"andf=dthen1070
1050 ife$="b"andg=ethen1070
1060 iff=dandg=ethen1070:else990
1070 print"[CTRL-9]";x;"[CTRL-0]";tab(6
     )b$(x);tab(32);:printusing"####.##
     ";c(x)
1080 iff>2thenv=v+c(x):elseu=u+c(x)
1090 e(m)=x:m=m+1
1100 h=h+1:ifh=ythen1130:else990
1110 ife$="d"thene=e+1:ife=13thene=1:d=
     d+1:ifd=15then1130
1120 ife$="d"thenx=0:goto990:else1130
1130 print"[2xCRSR-DOWN]totaal[SPACE]in
     komsten[SPACE]";:printusing"#####.
     ##";u
1140 print"[2xCRSR-DOWN]totaal[SPACE]ui
     tgaven[2xSPACE]";:printusing"#####
     .##";v
1150 print"[3xCRSR-DOWN]druk[SPACE]op[S
     PACE]een[SPACE]toets"
1160 gosub200:print"[SHIFT-CLR]":poke20
     22,0:print"[HOME]":return
1170 rem voor printen
1180 getkeyaa$:print"[SHIFT-CLR]";tab(1
     1)"[2xCRSR-DOWN]printen[7xSPACE]=[
     SPACE]p":printtab(11)"[2xCRSR-DOWN
     ]nieuwe[SPACE]keuze[2xSPACE]=[SPAC
     E]n"
1190 geta$:ifa$="n"then660
1200 ifa$="p"then1210:else1190
1210 print"[SHIFT-CLR][2xCRSR-DOWN]afdr
     ukken[SPACE]op[SPACE]a3[15xSPACE]=
     [SPACE]3"
1220 print"[2xCRSR-DOWN]afdrukken[SPACE
     ]op[SPACE]a4[15xSPACE]=[SPACE]4"
1230 print"[2xCRSR-DOWN]afdrukken[SPACE
     ]zonder[SPACE]stop[9xSPACE]=[SPACE
     ]z"
1240 geta$:ifa$="3"ora$="4"then1260
1250 ifa$="z"then1260:else1240
1260 print"[CRSR-DOWN]printer[SPACE]ger
     eed[SPACE]?[3xSPACE]druk[SPACE]op[
     SPACE]een[SPACE]toets":getkeyaa$
1270 print"[2xCRSR-DOWN]de[SPACE]lijst[
     SPACE]wordt[SPACE]afgedrukt"
1280 open1,4:m=0:k=0:ife$="a"ore$="c"th
     enprint#1,"rubriek:[SPACE]";a$(d)
1290 ife$="b"ore$="c"thenprint#1,"maand
     [2xSPACE]:[SPACE]";e
1300 ife$="d"thenprint#1,"gesorteerd[SP
     ACE]op[SPACE]rubriek[SPACE]en[SPAC
     E]maand"
1310 ife$="e"thenprint#1,"ongesorteerd"
1320 m=m+1:k=k+1:ife(m)=0then1390:goto6
     60
1330 print#1,e(m);chr$(32);:print#1,usi
     ng"########################";b$(
     e(m));
1340 print#1,using"#####.##";c(e(m)):e(
     m)=0
1350 ifa$="3"andk=80thengoto1370
1360 ifa$="4"andk=55thengoto1370:else13
     20
1370 print"[2xCRSR-DOWN]verwissel[SPACE
     ]papier[SPACE]-[2xSPACE]druk[SPACE
     ]op[SPACE]een[SPACE]toets";:k=0:ge
     tkeyaa$
1380 printchr$(27);"d";:goto1320
1390 print#1,"totaal[SPACE]inkomsten[SP
     ACE]";u;"[SPACE]totaal[SPACE]uitga
     ven[SPACE]";v:close1:goto660
1400 rem cassette en disk
1410 print"[2xCRSR-DOWN]welke[SPACE]naa
     m[SPACE]?[2xCRSR-DOWN]":inputam$
1420 ifa$="c"then1430:else1460
1430 print"[SHIFT-CLR][2xCRSR-DOWN]is[S
     PACE]de[SPACE]cassetterecorder[SPA
     CE]in[SPACE]orde[SPACE]?":print"[2
     xCRSR-DOWN]druk[SPACE]op[SPACE]een
     [SPACE]toets":getkeyaa$
1440 open1,1,0,am$:gosub1490:close1
1450 open2,1,0,"cij"+am$:gosub1510:clos
     e2:goto1650
1460 print"[SHIFT-CLR][2xCRSR-DOWN]is[S
     PACE]de[SPACE]diskdrive[SPACE]in[S
     PACE]orde[SPACE]?":print"[2xCRSR-D
     OWN]druk[SPACE]op[SPACE]een[SPACE]
     toets":getkeyaa$
1470 open1,8,2,"0:"+am$+",s,r":gosub149
     0:close1
1480 open2,8,2,"0:cij"+am$+",s,r":gosub
     1510:close2:goto1650
1490 x=1
1500 input#1,b$(x):os=st:ifos=0thenx=x+
     1:goto1500:elsereturn
1510 x=1
1520 input#2,c(x):os=st:ifos=0thenx=x+1
     :goto1520:elsereturn
1530 print"[2xCRSR-DOWN]welke[SPACE]naa
     m[SPACE]?[2xCRSR-DOWN]":inputam$
1540 ifa$="c"then1550:else1580
1550 print"[SHIFT-CLR][2xCRSR-DOWN]is[S
     PACE]de[SPACE]cassetterecorder[SPA
     CE]in[SPACE]orde[SPACE]?":print"[2
     xCRSR-DOWN]druk[SPACE]op[SPACE]een
     [SPACE]toets":getkeyaa$
1560 open1,1,1,am$:gosub1610:close1
1570 open2,1,1,"cij"+am$:gosub1630:clos
     e2:goto1650
1580 print"[SHIFT-CLR][2xCRSR-DOWN]is[S
     PACE]de[SPACE]diskdrive[SPACE]in[S
     PACE]orde[SPACE]?":print"[2xCRSR-D
     OWN]druk[SPACE]op[SPACE]een[SPACE]
     toets":getkeyaa$
1590 open1,8,2,"@0:"+am$+",s,w":gosub16
     10:close1
1600 open2,8,2,"@0:cij"+am$+",s,w":gosu
     b1630:close2:goto1650
1610 x=1
1620 print#1,b$(x):x=x+1:ifx=ythenretur
```

```
      n:else1620
1630  x=1
1640  print#2,c(x):x=x+1:ifx=ythenx=x-1:
      return:else1640
1650  y=x+1:goto660
1660  datasalaris,overige ink.,hypotheek
      ,telefoon,belastingen,verzekeringe
      n
1670  dataziektekosten,energie,kinderen,
      giften,boeken,video/geluid,rente,d
      iversen
1680  end
1690  rem voor opnieuw starten met behou
      d van data = goto 1700
1700  y=x:goto660
1702 rem voor meer/minder of andere pos
      ten de volgende wijzigingen aanbre
      ngen
1705 rem data aanpassen in regel 1660-1
      670
1710  rem wijzigingen aanbrengen in rege
      150-230-620-410-960-1080en tussen
       230tot 330
7095 rem hulpregel voor korrektie poke
      tijdens programmeren
8000  poke2021,24:poke2022,0:poke2025,0:
      poke2023,0
```

** EINDE LISTING boekhouding **



```
regel 10    171
regel 20    45
regel 30    18
regel 40    130
regel 50    17
regel 60    60
regel 70    180
regel 80    61
regel 90    230
regel 100    182
regel 110    170
regel 120    134
regel 130    169
regel 140    194
regel 150    194
regel 160    11
regel 170    245
regel 180    11
regel 190    206
regel 200    11
regel 210    161
regel 220    124
regel 230    7
regel 240    192
regel 250    189
regel 260    109
regel 270    2
regel 280    185
regel 290    193
regel 300    23
regel 310    109
regel 320    117
regel 330    149
regel 340    107
regel 350    162
regel 360    58
regel 370    142
regel 380    182
regel 390    190
regel 400    137
regel 410    96
regel 420    230
regel 430    75
regel 440    222
regel 450    234
regel 460    75
regel 470    57
regel 480    75
regel 490    225
regel 500    37
regel 510    116
regel 520    169
regel 530    162
regel 540    50
regel 550    35
regel 560    221
regel 570    165
regel 580    118
regel 590    124
regel 600    208
regel 610    156
regel 620    3
regel 630    176
regel 635    120
regel 640    2
regel 650    189
regel 660    54
regel 670    50
regel 680    216
regel 690    69
regel 700    71
regel 710    253
regel 720    170
regel 730    108
regel 740    232
regel 750    14
regel 760    49
regel 770    43
regel 780    134
regel 790    190
regel 800    80
regel 810    230
regel 820    3
regel 830    201
regel 840    114
regel 850    244
regel 860    156
regel 870    216
regel 880    116
regel 890    118
regel 900    133
regel 910    78
regel 920    112
regel 930    171
regel 940    36
regel 950    65
regel 960    142
regel 970    125
regel 980    229
regel 990    207
regel 1000    197
regel 1010    229
regel 1020    212
regel 1030    158
regel 1040    133
regel 1050    136
regel 1060    212
regel 1070    191
regel 1080    80
regel 1090    78
regel 1100    82
regel 1110    183
regel 1120    72
regel 1130    124
regel 1140    40
regel 1150    76
regel 1160    174
regel 1170    245
regel 1180    169
regel 1190    183
regel 1200    123
regel 1210    176
regel 1220    31
regel 1230    73
regel 1240    8
regel 1250    134
regel 1260    179
regel 1270    59
regel 1280    26
regel 1290    121
regel 1300    169
regel 1310    160
regel 1320    225
regel 1330    215
regel 1340    8
regel 1350    40
regel 1360    0
regel 1370    9
regel 1380    161
regel 1390    232
regel 1400    169
regel 1410    102
regel 1420    114
regel 1430    88
regel 1440    7
regel 1450    85
regel 1460    75
regel 1470    83
regel 1480    179
regel 1490    59
regel 1500    120
regel 1510    59
regel 1520    88
regel 1530    102
regel 1540    120
regel 1550    88
regel 1560    2
regel 1570    89
regel 1580    75
regel 1590    146
regel 1600    251
regel 1610    59
regel 1620    176
regel 1630    59
regel 1640    8
regel 1650    157
regel 1660    167
regel 1670    85
regel 1680    128
regel 1690    247
regel 1700    194
regel 1702    252
regel 1705    17
regel 1710    178
regel 7095    252
regel 8000    200
```

PRINT OUT Amiga met Rubik's Clock

Omdat er in de vorige rubriek de zetduivel ons de nodige parten heeft gespeeld, herhalen we hier nogmaals de Amiga listing uit het vorige nummer.

Rubiks Clock

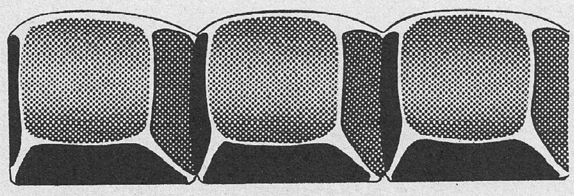
Het uurwerk dat waarschijnlijk de mensheid de laatste jaren het meeste heeft bezig gehouden in waarschijnlijk die van de Heer Rubik. Maar nu dit programma is gepubliceerd doet ook M.J. Kemps, de maker van deze computerversie van deze klok een goede gooi. Het programma is volledig muis gestuurd door middel van de pull-down menu's.

Let op de [RETURN] tussen de grote haken niet intypen maar uitvoeren. Dus gewoon op de return-toets drukken. Verder alles achter elkaar doortikken.

```
'Rubiks Clock [RETURN]
'door M.J. Kemps uit Schiedam [RETURN]
[RETURN]
CLEAR ,50000& [RETURN]
[RETURN]
Pointer: [RETURN]
FOR x=0 TO 63:READ y:POKE
20212+x,y:NEXT x [RETURN]
DATA 254,0,0,120,252,0,120
,248,248,0,113,199,240,0,97,134,224,0,65,1
28,192,6,1,128,128,7,1,192,0,0,0,248,0,0,0
,120,0,0,0,0,0,0,162,230,0,0,178,137,0,0,1
70,201,0,0,170,137,0,0,166,137,0,0,162,134
[RETURN]
[RETURN]
Startup: [RETURN]
RANDOMIZE TIMER [RETURN]
pi=3.1415926536#  [RETURN]
DIM DrawClock(6024),Hour(2,12),Button(4)
[RETURN]
clockside=1:Button(1)=1:Button(2)=1:Button
(3)=1:Button(4)=1 [RETURN]
SCREEN 2,320,216,3,1:WINDOW 2,"Rubiks
Clock",,16,2 [RETURN]
PALETTE 0,.4,.4,.4 [RETURN]
PALETTE 1,1,1,1 [RETURN]
PALETTE 2,1,0,0 [RETURN]
PALETTE 3,0,1,0 [RETURN]
PALETTE 4,1,1,0 [RETURN]
PALETTE 5,0,.5,1 [RETURN]
PALETTE 6,0,0,0 [RETURN]
PALETTE 7,0,1,1 [RETURN]
PALETTE 17,1,0,0 [RETURN]
PALETTE 18,0,0,.7 [RETURN]
PALETTE 19,1,1,1 [RETURN]
MENU OFF [RETURN]
MENU 1,0,1,"Programm":MENU 1,1,1,"Info
":MENU 1,2,1,"Quit     " [RETURN]
MENU 2,0,1,"Options":MENU
2,1,1,"Shuffle":MENU
2,2,1,"Instructions" [RETURN]
MENU 3,0,1,"":MENU 4,0,1,"":MENU
5,0,1,"" [RETURN]
ON MENU GOSUB Checkmenu: [RETURN]
[RETURN]
draw: [RETURN]
CIRCLE (206,54),33,6,,,1:PAINT
(206,54),6 [RETURN]
CIRCLE (114,54),33,6,,,1:PAINT
(114,54),6 [RETURN]
CIRCLE (114,146),33,6,,,1:PAINT
(114,146),6 [RETURN]
CIRCLE (206,146),33,6,,,1:PAINT
(206,146),6 [RETURN]
FOR a=0 TO 2*pi STEP pi/6 [RETURN]
x=160+95*COS(a) [RETURN]
y=100+95*SIN(a) [RETURN]
AREA (x,y) [RETURN]
NEXT a [RETURN]
COLOR 5,0 [RETURN]
AREAFILL  [RETURN]
CIRCLE(160,100),15,1,,,1:CIRCLE(160,82),
1,4,,,1:PAINT(160,82),4 [RETURN]
CIRCLE(114,54),15,1,,,1:CIRCLE(114,36),1,4
,,,1:PAINT(114,36),4 [RETURN]
CIRCLE(160,54),15,1,,,1:CIRCLE(160,36),1,4
,,,1:PAINT(160,36),4 [RETURN]
CIRCLE(206,54),15,1,,,1:CIRCLE(206,36),1,4
,,,1:PAINT(206,36),4 [RETURN]
CIRCLE(114,100),15,1,,,1:CIRCLE(114,82),1,
4,,,1:PAINT(114,82),4 [RETURN]
CIRCLE(206,100),15,1,,,1:CIRCLE(206,82),1,
4,,,1:PAINT(206,82),4 [RETURN]
CIRCLE(114,146),15,1,,,1:CIRCLE(114,128),1
,4,,,1:PAINT(114,128),4 [RETURN]
CIRCLE(160,146),15,1,,,1:CIRCLE(160,128),1
,4,,,1:PAINT(160,128),4 [RETURN]
CIRCLE(206,146),15,1,,,1:CIRCLE(206,128),1
,4,,,1:PAINT(206,128),4 [RETURN]
CIRCLE(137,77),5,4,,,1:PAINT(137,77),4
[RETURN]
CIRCLE(183,77),5,4,,,1:PAINT(183,77),4
[RETURN]
CIRCLE(137,123),5,4,,,1:PAINT(137,123),4
[RETURN]
CIRCLE(183,123),5,4,,,1:PAINT(183,123),4
[RETURN]
FOR clock=1 TO 12 [RETURN]
CIRCLE(160,100),3,1,,,1:PAINT(160,100),1
[RETURN]
x1=INT(3*SIN((9+clock)*pi/6)+.5):y1=INT(3*
COS((9+clock)*pi/6)+.5) [RETURN]
x2=INT(160+13*COS((clock+9)*pi/6)+.5):y2=I
NT(100+13*SIN((clock+9)*pi/6)+.5)
[RETURN]
AREA(160+x1,100-y1):AREA(x2,y2):AREA(160-x
1,100+y1):COLOR 1,5:AREAFILL [RETURN]
GET(145,85)-(175,115),DrawClock((clock-1)*
502) [RETURN]
PAINT(160,100),5 [RETURN]
NEXT clock [RETURN]
COLOR 4,0:LOCATE 23,1:PRINT " Side 1
":LOCATE 23,32:PRINT " Side 2 " [RETURN]
LINE(4,173)-(60,186),4,b:LINE(252,173)-(30
8,186),4,b [RETURN]
GOSUB Shuffle [RETURN]
MENU ON [RETURN]
[RETURN]
Puzzle: [RETURN]
```

```
IF MOUSE(0)=0 THEN Puzzle [RETURN]
x=MOUSE(1):y=MOUSE(2) [RETURN]
IF x>4 AND x<60 AND y>173 AND y<186
THEN clockside=1 [RETURN]
IF x>252 AND x<308 AND y>173 AND y<186
THEN clockside=2 [RETURN]
Change=0 [RETURN]
IF x>132 AND x<142 AND y>72 AND y<82
THEN Change=1 [RETURN]
IF x>178 AND x<188 AND y>72 AND y<82
THEN Change=2 [RETURN]
IF x>132 AND x<142 AND y>118 AND y<128
THEN Change=3 [RETURN]
IF x>178 AND x<188 AND y>118 AND y<128
THEN Change=4 [RETURN]
IF Change=1 AND clockside=1 OR Change=2
AND clockside=2 THEN
Button(1)=-Button(1)+3 [RETURN]
IF Change=2 AND clockside=1 OR Change=1
AND clockside=2 THEN
Button(2)=-Button(2)+3 [RETURN]
IF Change=3 AND clockside=1 OR Change=4
AND clockside=2 THEN
Button(3)=-Button(3)+3 [RETURN]
IF Change=4 AND clockside=1 OR Change=3
AND clockside=2 THEN
Button(4)=-Button(4)+3 [RETURN]
turn=0:direction=0 [RETURN]
IF x>81 AND y<54 AND x<y+60 THEN
turn=1:direction=-1 [RETURN]
IF x<114 AND y>21 AND x>y+60 THEN
turn=1:direction=1 [RETURN]
IF x>206 AND y>21 AND x<-y+260 THEN
turn=2:direction=-1 [RETURN]
IF x<239 AND y<54 AND x>-y+260 THEN
turn=2:direction=1 [RETURN]
IF x<114 AND y<179 AND x>-y+260 THEN
turn=3:direction=-1 [RETURN]
IF x>81 AND y>146 AND x<-y+260 THEN
turn=3:direction=1 [RETURN]
IF x<239 AND y>146 AND x>y+60 THEN
turn=4:direction=-1 [RETURN]
IF x>206 AND y<179 AND x<y+60 THEN
turn=4:direction=1 [RETURN]
IF clockside=2 THEN
direction=-direction:turn=3-turn [RETURN]
IF turn<1 THEN turn=turn+4 [RETURN]
IF Button(turn)=Button(1)THEN
Hour(1,1)=Hour(1,1)+direction [RETURN]
IF Button(turn)=Button(2)THEN
Hour(1,3)=Hour(1,3)+direction [RETURN]
IF Button(turn)=Button(3)THEN
Hour(1,7)=Hour(1,7)+direction [RETURN]
IF Button(turn)=Button(4)THEN
Hour(1,9)=Hour(1,9)+direction [RETURN]
IF Button(turn)=1 THEN [RETURN]
Hour(1,5)=Hour(1,5)+direction [RETURN]
IF Button(1)=1 OR Button(2)=1 THEN
Hour(1,2)=Hour(1,2)+direction [RETURN]
IF Button(1)=1 OR Button(3)=1 THEN
Hour(1,4)=Hour(1,4)+direction [RETURN]
IF Button(2)=1 OR Button(4)=1 THEN
Hour(1,6)=Hour(1,6)+direction [RETURN]
IF Button(3)=1 OR Button(4)=1 THEN
Hour(1,8)=Hour(1,8)+direction [RETURN]
ELSE [RETURN]
Hour(2,5)=Hour(2,5)-direction [RETURN]
IF Button(1)=2 OR Button(2)=2 THEN
Hour(2,2)=Hour(2,2)-direction [RETURN]
IF Button(1)=2 OR Button(3)=2 THEN
Hour(2,6)=Hour(2,6)-direction [RETURN]
IF Button(2)=2 OR Button(4)=2 THEN
Hour(2,4)=Hour(2,4)-direction [RETURN]
IF Button(3)=2 OR Button(4)=2 THEN
Hour(2,8)=Hour(2,8)-direction [RETURN]
END IF [RETURN]
GOSUB SetClocks [RETURN]
release: [RETURN]
IF MOUSE (0)=-1 THEN release [RETURN]
GOTO Puzzle      [RETURN]
[RETURN]
SetClocks: [RETURN]
IF clockside=1 THEN PALETTE
5,0,.5,1:ELSE PALETTE 5,0,0,1 [RETURN]
Hour(2,1)=12-Hour(1,3):Hour(2,3)=12-Hour(1
,1):Hour(2,7)=12-Hour(1,9):Hour(2,9)=12-Ho
ur(1,7) [RETURN]
FOR side=1 TO 2:FOR clock=1 TO 12
[RETURN]
IF Hour(side,clock)<1 THEN
Hour(side,clock)=Hour(side,clock)+12
[RETURN]
IF Hour(side,clock)>12 THEN
Hour(side,clock)=Hour(side,clock)-12
[RETURN]
NEXT clock,side [RETURN]
PUT(99,39),DrawClock((Hour(clockside,1)-1)
*502),PSET [RETURN]
PUT(145,39),DrawClock((Hour(clockside,2)-1
)*502),PSET [RETURN]
PUT(191,39),DrawClock((Hour(clockside,3)-1
)*502),PSET [RETURN]
PUT(99,85),DrawClock((Hour(clockside,4)-1)
*502),PSET [RETURN]
PUT(145,85),DrawClock((Hour(clockside,5)-1
)*502),PSET [RETURN]
PUT(191,85),DrawClock((Hour(clockside,6)-1
)*502),PSET [RETURN]
PUT(99,131),DrawClock((Hour(clockside,7)-1
)*502),PSET [RETURN]
PUT(145,131),DrawClock((Hour(clockside,8)-
1)*502),PSET [RETURN]
PUT(191,131),DrawClock((Hour(clockside,9)-
1)*502),PSET [RETURN]
IF Button(1)=1 AND clockside=1 OR
Button(2)=2 AND clockside=2 THEN
bcolor=4:ELSE bcolor=5 [RETURN]
CIRCLE(137,77),5,bcolor,,,1 [RETURN]
IF Button(2)=1 AND clockside=1 OR
Button(1)=2 AND clockside=2 THEN
bcolor=4:ELSE bcolor=5 [RETURN]
CIRCLE(183,77),5,bcolor,,,1 [RETURN]
IF Button(3)=1 AND clockside=1 OR
Button(4)=2 AND clockside=2 THEN
bcolor=4:ELSE bcolor=5 [RETURN]
CIRCLE(137,123),5,bcolor,,,1 [RETURN]
IF Button(4)=1 AND clockside=1 OR
Button(3)=2 AND clockside=2 THEN
bcolor=4:ELSE bcolor=5 [RETURN]
CIRCLE(183,123),5,bcolor,,,1 [RETURN]
RETURN [RETURN]
[RETURN]
Checkmenu: [RETURN]
menuid=MENU(0):menuitem=MENU(1) [RETURN]
```

```
ON menuid GOSUB Programm,Options
[RETURN]
RETURN [RETURN]
[RETURN]
Programm: [RETURN]
ON menuitem GOSUB Info,Quit [RETURN]
RETURN [RETURN]
[RETURN]
Info: [RETURN]
WINDOW 3,"Info",(74,50)-(248,138),0,2
[RETURN]
MENU OFF:COLOR 5,1:CLS [RETURN]
PRINT:PRINT "  Rubiks Clock  v1.0"
[RETURN]
COLOR 6,1:PRINT:PRINT "  Created in
1989 by" [RETURN]
COLOR 2,1:PRINT "       M.J. Kemps"
[RETURN]
COLOR 6,1:PRINT:PRINT "      Designed
for" [RETURN]
COLOR 5,1:PRINT "    Commodore INFO"
[RETURN]
COLOR 6,4:LOCATE 10,11:PRINT "OK"
[RETURN]
LINE (77,69)-(98,82),2,b [RETURN]
Checkmouse: [RETURN]
IF MOUSE(0)=0 THEN Checkmouse [RETURN]
IF MOUSE(3)<80 OR MOUSE(3)>95 THEN
Checkmouse [RETURN]
IF MOUSE(4)<72 OR MOUSE(4)>79 THEN
Checkmouse [RETURN]
COLOR 1,6:LOCATE 10,11:PRINT "OK"
[RETURN]
toetslos: [RETURN]
IF MOUSE(0)<>0 THEN toetslos [RETURN]
WINDOW CLOSE 3:MENU ON [RETURN]
RETURN [RETURN]
[RETURN]
Quit: [RETURN]
WINDOW 3,"Quit",(80,50)-(240,138),0,2
[RETURN]
MENU OFF:COLOR 5,1:CLS [RETURN]
LOCATE 3,4:PRINT "Are you sure ?"
[RETURN]
COLOR 6,4:LOCATE 9,3:PRINT " Quit
":LOCATE 9,13:PRINT "Cancel" [RETURN]
LINE (13,61)-(66,74),2,b:LINE
(93,61)-(146,74),2,b [RETURN]
Checkmouse2: [RETURN]
IF MOUSE(0)=0 THEN Checkmouse2 [RETURN]
IF MOUSE(3)<64 AND MOUSE(3)>15 AND
MOUSE(4)>63 AND MOUSE(4)<72 THEN Quit1
[RETURN]
IF MOUSE(3)<144 AND MOUSE(3)>95 AND
MOUSE(4)>63 AND MOUSE(4)<72 THEN Cancel
[RETURN]
GOTO Checkmouse2 [RETURN]
Quit1: [RETURN]
COLOR 1,6:LOCATE 9,3:PRINT " Quit "
[RETURN]
wacht1: [RETURN]
IF MOUSE(0)<>0 THEN wacht1 [RETURN]
WINDOW CLOSE 3:WINDOW CLOSE 2:SCREEN
CLOSE 2 [RETURN]
MENU RESET [RETURN]
END [RETURN]
Cancel: [RETURN]
COLOR 1,6:LOCATE 9,13:PRINT "Cancel"
[RETURN]
wacht2: [RETURN]
IF MOUSE(0)<>0 THEN wacht2 [RETURN]
WINDOW CLOSE 3:MENU ON [RETURN]
RETURN [RETURN]
[RETURN]
Options: [RETURN]
ON menuitem GOSUB Shuffle,Instructions
[RETURN]
RETURN [RETURN]
[RETURN]
Shuffle: [RETURN]
FOR side=1 TO 2:FOR clock=1 TO 12
[RETURN]
Hour(side,clock)=INT(12*RND(1)+1)
[RETURN]
NEXT clock,side [RETURN]
GOSUB SetClocks [RETURN]
RETURN [RETURN]
[RETURN]
Instructions: [RETURN]
WINDOW 4,"instructions",,0,2 [RETURN]
MENU OFF:COLOR 7,6:CLS [RETURN]
PALETTE 2,.8,0,.8:a=MOUSE(0) [RETURN]
PRINT :PRINT "All you have to do is to
get all clocks" [RETURN]
PRINT "at 12 o'clock by turning the
wheels and" [RETURN]
PRINT "changing the buttons." [RETURN]
PRINT :PRINT "Don't forget the other
side!" [RETURN]
PRINT:COLOR 2,6 [RETURN]
PRINT "Control:" [RETURN]
PRINT :PRINT "Changing Buttons:
";:COLOR 3,6:PRINT "Simply klick on
them" [RETURN]
COLOR 2,6:PRINT :PRINT "Turning wheels:
";:COLOR 3,6:PRINT "Klick on the side
of" [RETURN]
PRINT "                    the direction
you" [RETURN]
PRINT "                    want to turn"
[RETURN]
COLOR 2,6:PRINT :PRINT "Change side:
";:COLOR 3,6:PRINT "Klick on Side X"
[RETURN]
COLOR 4,6:LOCATE 20,10:PRINT "Press
left mousebutton" [RETURN]
klick1: [RETURN]
IF MOUSE(0)=0 THEN klick1 [RETURN]
WINDOW CLOSE 4:PALETTE 2,1,0,0:MENU ON
[RETURN]
RETURN [RETURN]
```



Foutjes in Commodore Info nr. 4

Helaas is in nummer vier van deze jaargang een paar keer een voor ons 'klassieke' fout geslopen. Enkele keren zijn > en < tekens weggevallen. Een aantal bellers van de listingtelefoon (maandagavond van 17.00-21.00 uur op nr. 02155-25162, NIET IN DE MAAND AUGUSTUS IN VERBAND MET VAKANTIE) maakte ons daarop ook attent, sommigen hadden ook zelf de oplossing al gevonden. Het gaat op de pagina's 72 en 73 om de volgende listings:

1.
```
GOSUB seconden : PRINT  USING
"#####.####"; sec#
seconden:
sec#=TIMER
IF VAL(LEFT$(TIME$,2))>17 AND
sec#<20865# THEN sec#=sec#+65536#
RETURN
```

2.
verschil:

```
s3#=s2#-s1#
IF s3#<0 THEN s3#=s3#+20864#
ELSE RETURN
IF s3#<0 THEN s3#=s3#+44672#
RETURN
```

3.
```
CALL seconden(s1#)
---------
---------
---------
CALL seconden(s2#)
s3#=s2#-s1# : IF s3#<0 THEN s3#=
s3#+86400#
PRINT USING "#####.####";
s3#
---------
---------
END

SUB seconden(s#) STATIC
s#=TIMER
IF VAL(LEFT$(TIME$,2))>17
AND s#<20865# THEN s#=s#+65536#
END SUB
```

Missers

In ons vorige nummer hebben we het programma Golf 64 geplaatst, hierin komt een raar teken voor een soort "kommaatje" boven aan de regel. Dit moet het @ (apestaartje) zijn. Dus overal in de listing dit teken wijzigen in @

In het programma Woordtraining komen een paar verkeerde woorden voor, U moet hier vervangen:

```
pudef    door safe
trap     door pause
dcclear  door color
```

Staat er dus trap2 dan maakt U daarvan pause2, dit moet door de gehele listing gebeuren.

R.G.

De informatierubriek over het populaire besturingssysteem GEOS onder redactie van Bert Venema. Lezers worden van harte uitgenodigd om hun vragen, tips en trucs op te sturen.

Geos INFO

Mail

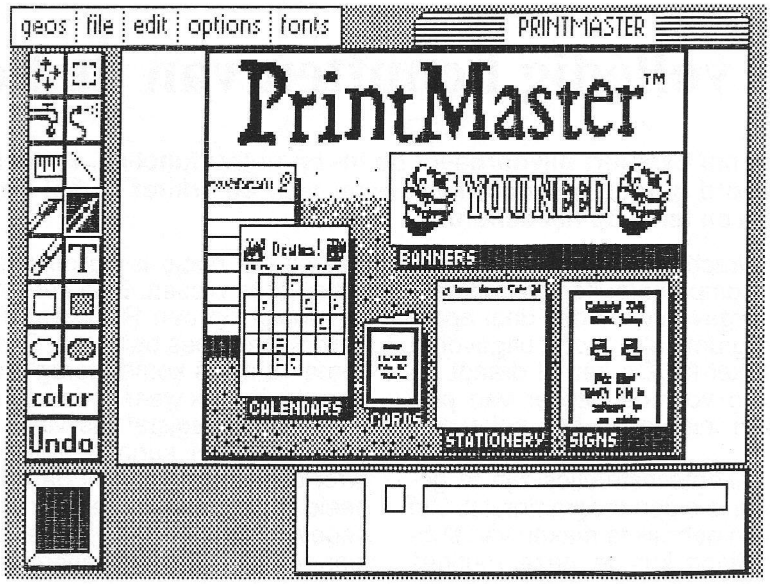
Van Ronald de Man uit Strijen ontvingen we een brief met daarin de mededeling dat hij een diskette had opgestuurd met een aantal programma's die zouden werken onder het GEOS besturingssysteem. Helaas hebben wij van deze diskette nog niets vernomen, maar we kunnen onze nieuwsgierigheid maar nauwelijks bedwingen. Vandaar ons verzoek om deze diskette opnieuw aan ons op te sturen. We zijn namelijk zeer benieuwd in hoeverre Nederlandse programmeurs erin slagen om programmatuur voor GEOS te kunnen ontwikkelen. Nog maar al te vaak moeten verschillende programma's uit het buitenland geïmporteerd worden. Indien er meer mensen zijn die in het bezit zijn van programmatuur, werkend onder GEOS, laat het ons dan even weten.

Printer oplossing

Een tijdje terug werden wij (wederom) eens geraadpleegd omtrent een printerprobleem voor de **STAR NX-10C** printer. Nu ontvingen wij laatst een reactie van Frans Rochette uit Heerlen, die ons kon melden dat er al sinds de 1.3 versie van GEOS een public domain diskette in de omloop is, die de gezochte printerdriver bezit. Indien er mensen zijn die deze printerdriver zoeken kunnen ze een briefje schrijven naar Frans Rochette, Marcellusstraat 12 6417 TK Heerlen. Wellicht is er een oplossing voor uw printerprobleem te vinden.

GEOS in een cartridge

Roger Popken uit Stadskanaal vindt het nogal irritant dat files in GEOS steeds bijgeladen moeten worden. Hij vindt dat dit in de eerste plaats nogal vertragend werkt en verder denkt hij dat de diskdrive hiervan ook te lijden heeft. Hij vraagt derhalve, waarom de GEOS Kernal en de DeskTop niet in een cartridge kunnen worden geplaatst, opdat het een en ander aanzienlijk versneld kan worden. Nou Roger, om eerlijk te zijn weten we niet waarom Berkeley Softworks dit tot op heden nog niet heeft gedaan, maar het is wel zo dat de GEOS-programmatuur nogal omvangrijk is en daarom niet in zijn geheel in een 32 Kbyte ROM gezet kan worden. Ik weet uit ervaring dat er enkele hobbyisten zijn die een gedeelte van het GEOS besturingssysteem in een ROM cartridge hebben geplaatst, maar men heeft het niet ontworpen om dit commercieel aan te pakken. Al met al is het idee niet slecht, maar ook niet echt nieuw meer.



GEOS netwerk???

Uit Döbra, Namibia ontvingen we een brief van dhr. A. van Dun. Hij houdt zich bezig met het opleiden van studenten. Hiervoor maakt hij gebruik van een achttal computers, welke zijn aangesloten op één 1541 diskdrive d.m.v. een VIC-switch. Hierdoor beschikt hij over een soort netwerk, waarmee de acht computers geladen kunnen worden. Alleen met GEOS komt hij voor problemen te staan. Hij kan steeds slechts één computer laden met GEOS, waarna de VIC-switch stopt. Alleen wanneer hij de drive steeds gereset wordt kan hij verder laden, hetgeen resulteert in dat alleen de laatst geladen computer werkt. Gezien het feit dat de computerhandelaren in Windhoek (de plaats waar dhr. van Dun op aangewezen is) ook niet over veel expertise beschikken, wendt hij zich tot onze rubriek met de vraag of wij een oplossing voor dit probleem kennen. In Nederland is zoiets niet voor handen, maar vorig jaar nog maakte de president van Berkeley Softworks, Brian Dougherty, melding van een applicatie **GeoNet** genaamd. Deze applicatie verbindt een IBM PC met maar liefst 32 Commodores 64 en zelfs met een Apple II. Met **GeoNet** is het mogelijk om te communiceren tussen de computers onderling. Het is wellicht nog niet precies wat u zoekt, maar misschien komt het een beetje in de richting. Het enige wat u moet doen is een brief sturen naar Berkeley Softworks, 2150 Shattuck Avenue, Berkeley, CA 94704, United States Of America. Wellicht kunnen zij u verder helpen met uw probleem. Het enige advies dat ik u op dit moment kan geven, is de aanschaf van een tweetal extra diskdrives en zodoende meerdere C64's geschikt te maken voor gebruik met GEOS. We wensen u echter veel sterkte bij uw zware taak in het roerige Namibia.

Het Commodore 128 Operating System (OS) bestuurt direct of indirect alle functies van uw computer. Het besturingssyteem is ondergebracht in een chip met 16 KROM machinetaal instructies. Deze chip wordt ook wel KERNAL genoemd. De instructies vormen samen de routines die alle functies van de machine besturen.

Commodore 128 OS

Het volledig benutten van de kernal



De kernal bestuurt bijvoorbeeld de in- en outputfuncties, inclusief toetsenbord output, het sturen van tekst naar de printer en het tonen van graphics en tekst op het scherm.

Elke opdracht die wordt uitgevoerd door de computer (buiten de opdrachten die uitgevoerd worden door applicatie programma's), wordt uitgevoerd door de kernal. De kernal draagt tevens zorg voor de uitvoer van programma's die je in het geheugen laadt.
De 16 K kernalinstructies zijn te gebruiken in je eigen programmatuur. In plaats van gebruik te maken van dubbele routines kun je deze routines aanroepen vanuit de eigen routines. Het aanroepen geschiedt via de *kernal jump table* (kernal spring tabel). Deze jump table bestaat uit een aantal jumps die naar de desbetreffende kernal routines springen. Deze jump table is opgenomen in de kernal om te voorkomen dat bij eventuele latere veranderingen in de kernal er incompatibiliteit optreedt. Een voorwaarde hierbij is wel, dat de programmatuur netjes geprogrammeerd is, dat wil zeggen dat er gebruik moet worden gemaakt van de kernal jump table. Het is natuurlijk ook mogelijk om de jump table te omzeilen. Je kunt dan bijvoorbeeld het spoor van de jump volgen, en in het vervolg meteen naar de kernal routine springen. Dit heeft als voordeel dat het desbetreffende programma sneller wordt. Denk er wel om dat dit niet bij alle computers zal werken, want niet elke ROM is hetzelfde.

Kernalroutines aanroepen

Als er een subroutine uit de kernal moet worden aangeroepen, moet je eerst kijken of de routine geen parameters of waarden nodig heeft. Deze parameters moeten in de accumulator, X-, of Y-registers worden geladen. Er zijn zelfs kernal routines waarbij het nodig is de carry flag te zetten of te wissen. Elke kernal routine eindigt op een RTS, wat 'Return from Subroutines betekent. Sommige kernal routines komen terug met een waarde die ook weer in de accumulator, X-, of Y-registers te vinden zijn. Deze waarden kunnen bijvoorbeeld error codes zijn. Je kunt dan bijvoorbeeld controleren welke error er wordt gegeven. Je kunt op de volgende manier een kernal routine aanroepen:

° 1. Laadt in de accumulator, X-, of Y-register een waarde.

° 2. Geef de juiste configuratie.

° 3. Roep de kernal routine nu aan met een JSR instructie, wat. jump to subroutine betekent.

Het adres van de jumpvector kun je in de kernal jump tabel vinden die elders in deze tekst afgedrukt staat. We kunnen hiervan het volgende voorbeeld geven. Het startadres van de routine BOOT CALL begint op het adres $FF53. Op dit adres staat een JMP instructie, wat Jump betekent. Deze 'jump' springt dan weer naar de desbetreffende subroutine van de kernal.

° 4. Als de kernal routine wordt beëindigd met behulp van een RTS kunnen er geheel andere waarden in de accumulator, X-, of Y_registers staan. Als er dus een error aanwezig is kun je door middel van het programma controleren welke error het is. Dit controleren gebeurt natuurlijk na de JSR instructie.

De 128 systeem vectors

De commodore 128 beschikt over een aantal vectoren die zowel in de ROM als in het RAM geheugen terug te vinden zijn. Deze zogenaamde system vectors dragen er zorg voor dat bij een eventuele interrupt altijd de interrupt routines van de kernal bereikbaar zijn, ongeacht de configuratie van het geheugen. Deze vectoren worden normaal niet door de computergebruiker benut, omdat het hier gaat om interrupt routines. Maar toch is het interessant genoeg om even bij deze vectoren stil te staan. Het is mogelijk met behulp van deze vectoren een hardware reset op te vangen of een stop/restore te onderscheppen. We kennen de volgende systeem vectoren:

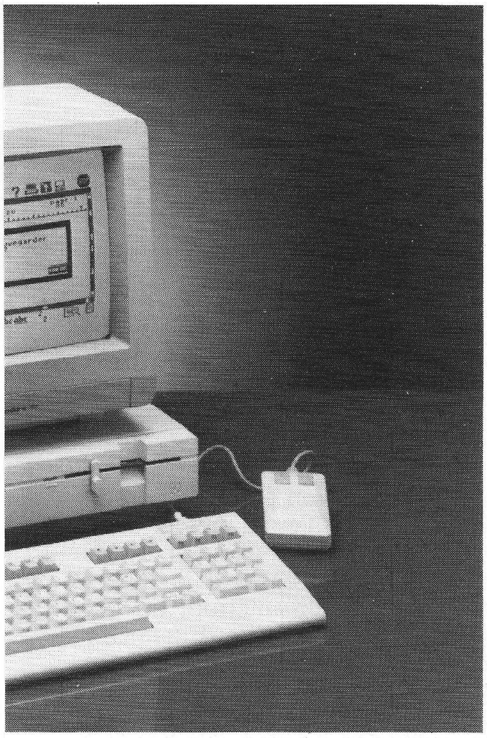
```
$FFF8/$FFF9   SYSTEM
$FFFa/$FFFB   NMI
$FFFC/$FFFD   RESET
$FFFE/$FFFF   IRQ/BRK
```

- SYSTEM

De SYSTEM vector en de bijbehorende 'CBM' key maken het mogelijk een hardware reset te onderscheppen. We bedoelen hiermee dat het moge-

lijk is de computer, na een reset, naar een programma van de gebruiker te laten springen. De SYSTEM vector is in tegenstelling tot de andere systeem vectoren alleen in bank 1 aanwezig en ziet er als volgt uit:

```
$FFF5 'C'   ($43)
$FFF6 'B'   ($42)
$FFF7 'M'   ($4d)
$FFF8 SYSTEM low vector
$FFF9 SYSTEM high vector
```



Als de computer nu gereset wordt, dan zal na verloop van tijd gecontroleerd worden op de 'CBM' key. Als deze niet gevonden wordt zal de rest van de reset procedure uitgevoerd worden. Maar als de 'CBM' key wel gevonden wordt, zal er naar het adres worden gesprongen die de SYSTEM vector aanwijst. Meestal is het dan alleen nog maar nodig een IOINIT uit te voeren.
Een voorbeeld. We nemen even aan dat er een programma staat op adres $2000. Dit programma moet na een reset opgestart worden. Om dat te bereiken moet de 'CBM' key en de SYSTEM vector gezet worden.

```
$fff5 43 42 4d 00 20
```

Het is verstandig het programma op $2000 nu te laten beginnen met een IOINIT (jsr $FF84).

- NMI

NMI staat voor Non-Maskable Interrupt, hetgeen vrij vertaald betekent, dat deze interrupt niet 'geblokkeerd' kan worden door een bit in één of ander register. Onder normale omstandigheden zijn er twee mogelijke NMI bronnen, namelijk de RESTORE-toets en de RS-232 poort. Bij een NMI interrupt zal het besturingssysteem van de computer er voor zorgen dat geen IRQ interrups meer toe worden gelaten. Vervolgens zullen de registers van de processor en de geheugenconfiguratie op stack worden gezet, de systeemconfiguratie (ROM, I/O, RAM 0) worden ingeschakeld en volgt er een indirecte sprong op de NMI_vector op $0318/$0319. Deze NMI_vector verwijst, mits deze niet veranderd is, naar het adres $FA40 waar het Interrupt Control Register van CIA-2 wordt gecontroleerd. Als de NMI interrupt afkomstig is van de timer A, wordt er verder gesprongen naar de RS-232 routines. Als de interrupt niet afkomstig is van timer A, dan wordt aangenomen dat de interrupt afkomstig is van de RESTORE-toets. De volgende stap is het controleren op de STOP-toets. Als deze ook is ingedrukt zullen alle indirecte systeem vectoren ($0300 tot en met $0320) weer terug worden gezet en worden IOINIT en CINT aangeroepen. Vervolgens wordt er naar het adres gesprongen dat aangewezen wordt door de SYSTEM_VECTOR (denk er wel om niet verwarren met de SYSTEM vector). De SYSTEM_VECTOR is te vinden op het adres $0A00/$0A01. Via deze vector wordt er naar $FF33 gesprongen waar de registers van de processor en de geheugenconfiguratie weer terug worden gezet. Er zijn dus twee manieren voor een programmeur om een NMI interrupt te onderscheppen, namelijk de NMI_vector op $0318/$0319 of de SYSTEM_VECTOR (in het geval van een STOP/RESTORE) op $0A00/$0A01. Bijvoorbeeld, ons programma staat nog steeds op adres $2000 en moet nu ook weer opnieuw worden opgestart als een lolbroek het probeert te onderbreken met een STOP/RESTORE. Het enige wat er moet gebeuren bij het opstarten van het programma is, dat òf de SYSTEM_VECTOR op $0A00/$0A01 òf de NMI_vector op $0318/$0319 'omgebogen' moet worden naar het begin van het programma. Dus:

```
$0A00 00 20
```
of:
```
$0318 00 20.
```

- RESET

De reset wordt geactiveerd wanneer het reset signaal laag wordt. Dit signaal is laag als de computer aangezet wordt en wordt ook laag gemaakt door op de reset knop te drukken. Het signaal heeft niet alleen invloed op de processor maar ook op de meeste I/O electronica. Wanneer de computer aangezet of gereset wordt, neemt de Z80 processor in eerste instantie de besturing van het systeem op zich. De 8502 processor wordt zolang in een 'wait state' (lees wachtfase) gezet. Wanneer de 8502 processor uiteindelijk opgestart wordt, wordt altijd de kernal routine START geactiveerd. Deze routine voert de volgende acties uit:

° 1. Neemt de SYSTEM configuratie aan. (ROM's, I/O).
° 2. Verhindert IRQ interrupts.
° 3. Reset processor stack pointer.
° 4. Zet de decimaal mode uit.
° 5. Initialiseert de MMU.
° 6. Installeert de kernal RAM code.

Tot zover is er geen mogelijkheid voor de gebruiker om eigen routines op te starten. De volgende twee routines die aangeroepen worden vanuit de kernal START routine kijken wel of er eventueel een programma van de gebruiker aanwezig is.

° 7. SECUREL: controleert de SYSTEM vector.
° 8. POLL: controleert of er een ROM cardridge aanwezig is.

POLL kijk of er een C64 cartridge geïnstalleerd is. Deze cartridges zijn te herkennen doordat òf het GAME òf het EXROM signaal laag is. Als dit het geval is wordt de kernal routine GO64 uitgevoerd. Het controleren op C64 cartridges is op dit moment eigenlijk niet meer nodig, omdat dit al gedaan is door de Z80 processor. Hierna controleert POLL of er een C128 cartridge of ROM's aanwezig zijn. Deze cartriges of ROM's worden herkend aan de zogenaamde 'CBM KEY'. Het gehele formaat is:

```
$x000 koude start doorgang
$x003 warme start doorgang
$x006 ID. ($01-$FF)
$x007 CBM key string
```

Met de x worden in dit geval de getallen $8--- of $C--- bedoeld.

Vervolgens worden de volgende routines afgehandeld.

° 9. IOINIT: initialiseert I/O devices.

° 10. Controle op STOP en commodore toets.

° 11. RAMTAS: initialiseert het RAM geheugen.

° 12. RESTOR: initialiseert indirecte vectoren.

° 13. CINT: initialiseert video chips.

° 14. IRQ wordt weer toegestaan.

° 15. controle wordt overgedragen aan òf de Basic òf GO64 routines òf aan de machinetaal monitor.

IOINIT is wellicht de belangrijkste stap die er in de RESET routine aangeroepen wordt. Deze routine initialiseert namelijk de beide CIA's. De CIA's beheren de timers, toetsenbord, de seriële poort, de user poort en de cassette poort. Ook initialiseert IOINIT drie 8502 poorten, namelijk het toetsenbord, cassette en de VIC bank. IOINIT onderscheidt een PAL systeem van NTSC systeem en zet vervolgens PALCNT ($0A03) als dat nodig is. Vervolgens worden de VIC, SID en de 8563 video chip geïnitialiseerd, inclusief het kopiëren van de karakterset naar de video ram van de 8563 video chip als dat nodig is. IOINIT kan door de gebruiker aangeroepen worden via de jump table.

Tijdens de uitvoering van al deze initialisaties wordt gecontroleerd of er bepaalde toetsen ingedrukt zijn (Commodore toets en de STOP toets). Als de Commodore toets is ingedrukt geeft dit aan dat er voor het C64 systeem wordt gekozen. Eigenlijk is dit een extra controle, aangezien de Z80 processor deze controle in het begin ook al eens had uitgevoerd. De andere toets waarop gecontroleerd wordt is de STOP toets. Met deze toets kan de gebruiker tijdens een reset (eigenlijk tijdens IOINIT) aangeven dat, als de reset uitgevoerd is, er gekozen moet worden voor de monitor. Als de STOP toets ingedrukt is zal de RAMTAS routine overgeslagen worden. Dit is erg handig omdat RAMTAS namelijk het geheugen initialiseert. Mocht het nu eens nodig zijn de computer te resetten, maar de gebruiker wil wel dat zijn programma in het geheugen blijft staan, dan moet dus de STOP toets ingedrukt worden tijdens de reset. RAMTAS kan net als IOINIT via de jump table aangeroepen worden. De volgende routine die aangeroepen wordt is de RESTOR routine. Deze routine initialiseert alle indirecte vectoren. Moeten in een programma een of meerde van deze vectoren veranderd worden, dan dient dit pas na de RESTOR te gebeuren. Ook de RESTOR routine kan via de jump table aangeroepen worden.
De laaste routine die in de hele reset procedure aangeroepen wordt is CINT. De CINT routine is de initialisatie routine voor de editor in zowel de 40 cls als in de 80 cls mode. Bij het aanroepen van deze routine worden beide modes voorbereid. De indirecte vectoren die nodig zijn voor de editor worden gezet, de programeerbare toetsen worden gezet en vervolgens wordt de 40/80 cls toets gecontroleerd en naar de gewenste video mode overgeschakeld. Ook CINT is weer via de jumptable aan te roepen. Als laatste wordt de IRQ weer toegestaan en wordt de controle overgedragen aan de Basic, de monitor of aan de C-64 mode.

-IRQ/BRK

Deze vector wordt elke keer genomen als de IRQ pin van de processor laag is of als de processor de BRK instructie uitvoert. Voor een goede werking van de IRQ moet deze 60 keer per seconde optreden (NTSC 60 Hz). Voor PAL systemen (50 Hz) zijn er een aantal aanpassingen nodig. Gewoonlijk is de VIC raster de bron voor een IRQ interrupt. Deze wordt tijdens een IOINIT even uitgeschakeld. Tijdens een IRQ of een BRK schrijft de kernal de huidige geheugenconfiguratie en het processor register op de stack. Hierna worden de ROM en het I/O gebied aangezet. De processor status op het tijdstip van de interrupt wordt dan van stack gehaald en aan de hand hiervan wordt bepaald of het hier gaat om een IRQ interrupt of een BRK. In de kernal wordt dit als volgt gedaan:

```
TSX   :haal de stack pointer.
LDA $0105,X :haal de
    processor status.
AND #$10     :en controleer
    de BRK flag.
```

Als de BRK flag is gezet, wordt de controle overgedragen aan de machinetaal monitor via de BRK indirecte vector $0316/$0317 die gewoonlijk naar de monitor wijst. Als de BRK flag niet is gezet is er dus een IRQ interrupt opgetreden en wordt er via de IRQ indirecte vector $0314/$0315 naar de IRQ routines in de kernal gesprongen. Nu worden de volgende stappen ondernomen:

° 1. IRQ's worden niet meer toegestaan.

° 2. EDITOR: split screen routines worden uitgevoerd.

° 3. EDITOR: de VIC raster IRQ wordt op nul gezet.

° 4. IRQ's worden weer toegestaan.

° 5. EDITOR: toetsenbord wordt gecontroleerd.

° 6. EDITOR: VIC cursor blink wordt uitgevoerd.

° 7. KRNAL: "jiffie" klok wordt behandeld.

° 8. KERNAL: cassette routines.

° 9: KERNAL: CIA-1 ICR wordt 'gereset'.

° 10: BASIC: sprites, geluid e.d. worden uitgevoerd.

° 11: RTI return from interrupt.

Zoals u ziet wordt de IRQ door de C128 intensief gebruikt. Erg belangrijk zijn de split screen routines. Om een tekst- en een grafisch scherm te combineren is het nodig deze routines niet 60 keer per seconde aan te roepen maar 120 maal per seconde. Alleen de 'hoofd'-interrupt voert alle 11 stappen uit. Maar als er gebruik wordt gemaakt van een split screen, zullen de routines die daarvoor nodig zijn een keer zovaak uitgevoerd worden. Om nu onderscheid te maken tussen een 'hoofd'-interrupt en een 'split screen' interrupt wordt in het geval van de 'hoofd'-interrupt de carry flag gezet. Tijdens een split screen moeten directe I/O acties naar de VIC chip vermeden worden. Als het toch nodig is gebruik te maken van de VIC en er wordt gebruik gemaakt van een split screen, dan dient deze eerst uitgeschakeld te worden via de GRAPHM flag ($00d8). Als deze flag op $ff wordt gezet dan worden alle editor acties uitgeschakeld. De editor is verantwoordelijk voor de invoer vanaf het toetsenbord. De routine SCNKEY wordt hier gebruikt. Belangrijk is dat deze routine twee maal een indirecte vector neemt, namelijk KEYLOG ($033a/$033b) en KEYPUT ($033c/$033d).
Het toetsenbord wordt via de CIA 1 (PRA,PRB), REG $47 van de VIC (voor het numeriek toetsenbord) en via de 8502 poort (bit 6 voor CAPS LOCK). De SCNKEY routine kan via de jump table opgeroepen worden. De kernal software klok wordt via de routine UDTIM bijgehouden. Ook deze routine heeft een jump in de jump table. De IRQ routine maakt als laatste een sprong naar de BASIC_IRQ ($4006) maar alleen als bit 0 van INIT_STATUS is gezet (dit is al eerder besproken). Ook nu geld, net als

C128 SYSTEEM VECTOREN

```
$FFFB (65531) SYSTEM          ;operating system vector
                               (RAM1)
$FFFA (65530) NMI             ;processor NMI vector
$FFFC (65532) RESET           ;processor RESET vector
$FFFE (65534) IRQ             ;processor IRQ/BRK vector
```

CBM STANDAARD KERNAL JUMP TABEL

```
$FF81 (65409) JMP CINT        ;init screen editor and
                               devices
$FF84 (65412) JMP IOINIT      ;init I/O devices
$FF87 (65415) JMP RAMTAS      ;init RAM and buffers
$FF8A (65418) JMP RESTOR      ;init indirect vectors (system)
$FF8D (65421) JMP VECTOR      ;init indirect vectors (user)
$FF90 (65424) JMP SETMSG      ;kernal messages on/off
$FF93 (65427) JMP SECND       ;serial: send SA after LISTN
$FF96 (65430) JMP TKSA        ;serial: send SA after TALK
$FF99 (65433) JMP MEMTOP      ;set/read top of system RAM
$FF9C (65436) JMP MEMBOT      ;set/read bottom of system RAM
$FF9F (65439) JMP KEY         ;scan keyboard
$FFA2 (65442) JMP SETTMO      ;(reserved)
$FFA5 (65445) JMP ACPTR       ;serial: byte input
$FFA8 (65448) JMP CIOUT       ;serial: byte output
$FFAB (65451) JMP UNTLK       ;serial: send untalk
$FFAE (65454) JMP UNLSN       ;serial: send unlisten
$FFB1 (65457) JMP LISTN       ;serial: send listen
$FFB4 (65460) JMP TALK        ;serial: send talk
$FFB7 (65463) JMP READSS      ;read I/O status byte
$FFBA (65466) JMP SETLFS      ;set channel LA, FA, SA
$FFBD (65469) JMP SETNAM      ;set filename pointers
$FFC0 (65472) JMP (IOPEN)     ;open logical file
$FFC3 (65475) JMP (ICLOSE)    ;close logical file
$FFC6 (65478) JMP (ICHKIN)    ;set input channel
$FFC9 (65481) JMP (ICKOUT)    ;set output channel
$FFCC (65484) JMP (ICLRCH)    ;restore default channels
$FFCF (65487) JMP (IBASIN)    ;input from channel
$FFD2 (65490) JMP (IBSOUT)    ;output to channel
$FFD5 (65493) JMP LOAD        ;load from file
$FFD8 (65496) JMP SAVE        ;save to file
$FFDB (65499) JMP SETTIM      ;set internal clock
$FFDE (65502) JMP RDTIM       ;read internal clock
$FFE1 (65505) JMP (ISTOP)     ;scan STOP key
$FFE4 (65508) JMP (IGETIN)    ;read buffered data
$FFE7 (65511) JMP (ICALL)     ;close all files and channels
$FFEA (65514) JMP UDTIM       ;increment internal clock
$FFED (65517) JMP SCRORG      ;get current screen window size
$FFF0 (65520) JMP PLOT        ;read/set cursor position
$FFF3 (65523) JMP IOBASE      ;read base address of I/O block
```

NIEUWE C128 KERNAL JUMP TABEL

```
$FF47 (65351) JMP SPIN SPOUT  ;setup fast serial ports for
                               I/O
$FF4A (65354) JMP CLOSE ALL   ;close all files on a device
$FF4D (65357) JMP DMA CALL    ;send command to DMA device
$FF50 (65360) JMP C64MODE     ;reconfigure system as a C64
$FF53 (65363) JMP BOOT CALL   ;boot load program from disk
$FF56 (65366) JMP PHOENIX     ;init function cardridges
$FF59 (65369) JMP LKUPLA      ;search tables for given LA
$FF5C (65372) JMP LKUPSA      ;search tables for given SA
$FF5F (65375) JMP SWAPPER     ;switch between 40 and 80
                               columns
$FF62 (65378) JMP DLCHR       ;init 80-col character RAM
$FF65 (65381) JMP PFKEY       ;program a function key
$ff68 (65384) JMP SETBANK     ;set abnk for I/O operations
$FF6B (65387) JMP GETCFG      ;lookup MMU data for given bank
$FF6E (65390) JMP JSRFAR      ;gosub in another bank
$FF71 (65393) JMP JMPFAR      ;goto another bank
$FF74 (65396) JMP INDFET      ;LDA (fetvec),Y from any bank
$FF77 (65399) JMP INDSTA      ;STA (stavec),Y to any bank
$FF7A (65402) JMP INDCMP      ;CMP (cmpvec),Y to any bank
$FF7D (65405) JMP PRIMM       ;print immediate utility
```

bij een split screen, dat als de Basic gebruik maakt van de VIC of de SID, dan dient de gebruiker (programmeur) met de nodige voorzichtigheid gebruik te maken van deze CHIP's. Het beste is gewoon eerst de desbetreffende Basic acties uit te schakelen alvorens gebruik te maken van de VIC of de SID. Het is mogelijk om via de IRQ vectoren een programma op te starten. Met behulp van het volgende voorbeeld kan dat worden gedaan:

```
SEI
LDA #$00
STA $0314
LDA #$20
STA $0315
CLI
RTS
```

Op deze manier kunnen we een programma opstarten op het adres $2000. Voordat het programma wordt opgestart worden de volgende stappen ondergaan:

° 1. De IRQ interrupt wordt niet meer toegelaten.

° 2. De low-nybble van het adres $2000 wordt op $0314 weggezet (dat zijn dus de twee nullen).

° 3. De Hi-nybble van het adres $2000 wordt op $0315 weggezet (dat is dus de 20)

° 4. Nu kan de IRQ interrupt weer worden toegelaten.

Hiernaast staat de JUMP TABLE:

Error

Zoals al eerder vermeld, kan er na een sprong op een kernal routine een waarde terugkomen die in een van de registers is geladen (A, X of Y). Als er een error aanwezig is tijdens het aanroepen van een kernal routine, wordt de carry bit van het status_register gezet en het nummer van de error in de accumulator geladen. Hieronder volgt een lijst van errors met hun waarde, die kunnen voorkomen bij het aanroepen van een kernal routine. Let er wel op dat deze errors niet bij alle kernal I/O routines voorkomen.

Nr.	betekenis
0.	Routine terminated by the STOP key
1.	Too many open files
2.	File already open
3.	File not open
4.	File not found
5.	Device not present
6.	File is not an input file
7.	File is not an output file

8. Filename is missing
9. Illigal device number
41. File read error

Kernal sprongen

Als een kernal routine wordt opgeroepen moet er ook op de juiste configuratie worden gelet. Bijvoorbeeld de juiste bank.
De jump tabel die hierboven staat wordt hieronder volledig besproken.

1. $FF81 CINT ;initialize screen editor and devices

VOORBEREIDING: geen
RESULTATEN:
register: A, X en Y wordt gebruikt
Geheugen: init editor RAM
init editor I/O
Flags : geen

Voorbeeld:

```
SEI
JSR CINT
CLI
```

CINT is de initialisatie routine van de editor. De routine werkt zowel in de 40 als in de 80 koloms mode. Cint zet de VIC bank en de VIC nybble bank, zet de karakterrom goed, reset het volume van de SID-chip en maakt de 40 en 80 kolomsschermen schoon. Het enige waar de CINT routine geen betrekking op heeft is de in- en output initialisatie. Deze zijn nodig voor de IRQ's (toetsscan, cursor blink, splitscherm modus, scherm achtergrond kleur enz. Dit alles zullen we nog wel bij 'IOINIT' tegenkomen. Omdat CINT de editor indirecte vectoren update, welke worden gebruikt tijdens het IRQ proces, moet deze eerst worden uitgezet voordat de CINT routine wordt aangeroepen. Dit staat ook in het voorbeeld. CINT benut de status byte INIT_STATUS ($A04) als volgt:

$A04 bit 6 = 0 - volledige initialisatie
bit 6 = 1 - gedeeltelijke initialisatie

CINT kan ook worden aangeroepen doormiddel van adres ($C000).

2. $FF84 IOINIT ;intialiseert in- en output devices

VOORBEREIDING: geen
RESULTATEN:
Register: A, X en Y wordt gebruikt
Geheugen: initialisatie in- en output
Flags : geen

Voorbeeld:

```
SEI
JSR $FF84
CLI
```

IOINIT initialiseert beide CIA's (timmers, toetsenbord, seriële poort, userpoort cassette) en de 8502 poort (toetsenbord, cassette, VIC bank). De VIC, SID en 8563 devices worden geïnitialiseerd, inclusief de 8563 karakter definitie (als dat nodig is). IOINIT benut de status byte INIT_STATUS ($A04) als volgt:

$0a04 bit 7 = 0 - volledige initialisatie
bit 7 = 1 - gedeeltelijke initialisatie (geen 8563 karakter initialisatie)

Je moet er wel zeker van zijn dat de IRQ is uitgeschakeld voordat je de IOINIT aanroept.

We houden er nu even mee op, maar we kunnen u mededelen dat dit niet het laatste van het Commodore 128 besturingssysteem is. Er zijn namelijk nog heel wat kernal routines die nog niet besproken zijn.

JOHAN & JOHAN

De bootsector van de Amiga is vooral bekend geworden door de virussen die zich erin nestelen. In dit artikel zullen we een hardwarematige oplossing geven, die aangeeft wanneer er in de bootsector wordt geschreven. Verder wordt er nog een voorbeeld gegeven van een programma dat de bootsector kan kopiëren.

De bootsector

Geschiedenis

De Zwitserse groep SCA (Swiss Cracking Association) kwam als eerste met een virus in de bootsector. Dit virus werd alleen actief na de reset, dan kopieerde het programma zichzelf naar de diskette in DF0:. Na een aantal keren resetten verscheen er op het beeldscherm een tekst. Dit was toen nog origineel. Later volgde er een ware stormvloed van virussen die kwalijker waren en minder origineel. De virussen leken allemaal op het SCA virus behalve dan dat er geen reset nodig was om ze actief te maken.

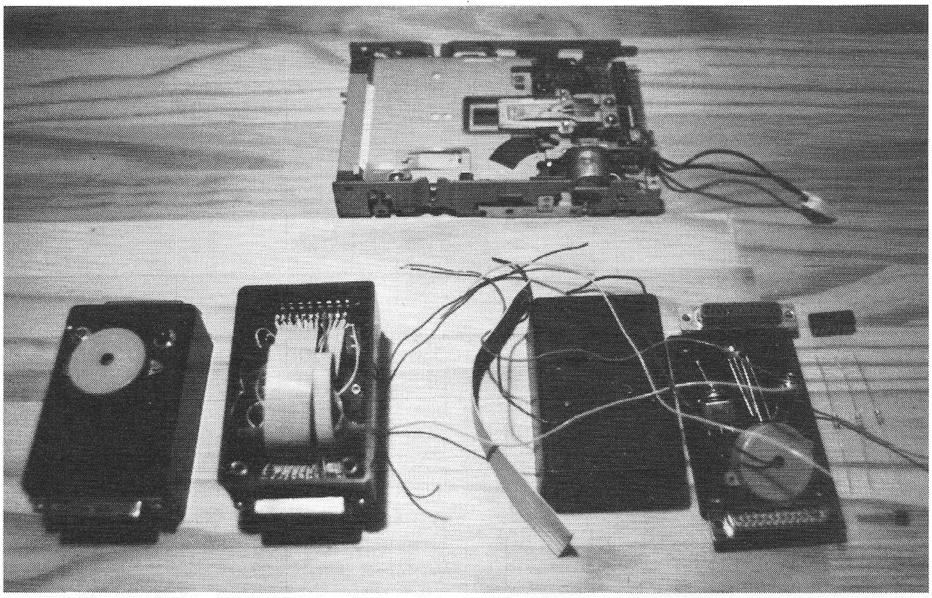
Eén van de kenmerken van virussen is dat bijvoorbeeld het scherm voor een aantal minuten een egale kleur aanneemt (Byte Bandit). Een ander virus geeft read/write errors op de diskette (DASA virus).

Werking

Een virus bevind zich hoog in het geheugen. De routine die normaal wordt uitgevoerd bij een reset, wordt door een virus veranderd. Bij een reset zal het virus dus niet uit het geheugen gaan, dit kan alleen gebeuren door de computer uit te zetten. Als er een diskette in de drive wordt gestopt gaat de diskdrive even draaien, op dit moment leest de amiga dan enkele gegevens over de diskette in. Deze staan in de bootsector (de eerste twee sectors van een diskette. De Amiga kijkt bijvoorbeeld of de aanwezige diskette wel een DOS-disk is. Als de Amiga deze gegevens uit de bootsector haalt zal een virus toegrijpen op de diskette (indien de diskette niet 'write protected' is).

Wat ertegen te doen

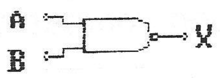
Er zijn vele hulpmiddelen in omloop om de virussen te bestrijden (de zogenaamde viruskillers). Zo zijn er bijvoorbeeld viruskillers die zich in de bootsector bevinden. Deze viruskillers zijn handig in het gebruik omdat men niet constant op virussen hoeft te controleren (zolang de viruskiller in de bootsector aanwezig is kan er geen virus in de bootsector staan). Hiermee hebben we dan meteen nog een



Op de voorgrond de Viruswarner (in en uitelkaar) achter een diskdrive

oplossing gevonden om virussen te vermijden. Men kan met een bootwriter zelf een tekst in de bootsector zetten. Zolang er bij het opstarten een tekst op het scherm verschijnt is er geen virus aanwezig. Bij beide manieren om virussen tegen te gaan is er een probleem. Als er wordt opgestart met een diskette met een virus erop en later wordt er gewisseld (het virus is dan in het geheugen aanwezig) en op deze diskette staat een speciale bootsector (viruskiller?) zal deze tijdens het initialiseren door het virus worden vernietigd.

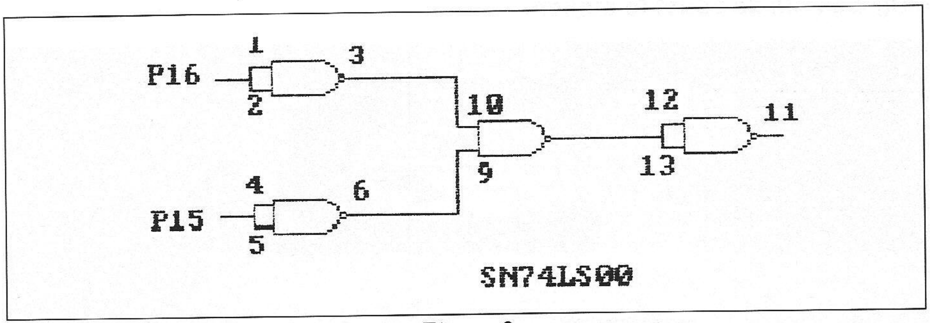
Een andere manier om de virussen te lijf te gaan is door alle diskettes een voor een te controleren (vernietigen). Hiervoor zijn verschillende public domain en dure commerciële programma's verkrijgbaar. Dit kan echter een tijdrovende bezigheid worden.
Beide manieren van bestrijden zijn tijdrovend en dienen voortdurend uitgevoerd te worden. Een goede oplossing is ons zelfbouw project. Deze maand is dit een viruswarner. Men hoeft hier nooit naar om te kijken omdat deze indien aangesloten altijd aanwezig is. Let wel, er wordt alleen aangegeven dat het virus zich aan het



A	B	X
L	L	H
H	L	H
L	H	H
H	H	L

Figuur 1

vermenigvuldigen is en er wordt niets tegen gedaan. Zo gauw men de zoemer hoort (sirene, zwaailicht) en het rode ledje aan gaat is er paniek in de tent omdat er dan een virus in de buurt is. Men dient dan met een hulpprogrammaatje het virus te verwijderen. (De TRISTAR viruskiller is aan te bevelen).



Figuur 2

De viruswarner

De viruswarner wordt op de diskdrive-connector aangesloten. Hij geeft een signaal als er in de bootsector wordt geschreven, ongeacht welke drive. Dit komt omdat de viruswarner continue de lijnen DKWE en TK0 controleert. wanneer de lijn TK0 een laag signaal (0 volt) heeft dan bevind de lees- en schrijf kop van de diskdrive zich op de plaats van track 0 (de plaats van de bootsector). Indien ook de lijn DKWE een laag signaal heeft dan wordt er informatie naar de diskdrive gestuurd en wordt er dus in de bootsector geschreven.

De hardware

Het belangrijkste onderdeel in de schakeling is IC 1, een SN74LS00. Dit IC bevat vier NAND poorten met elk twee ingangen. Als nu A en B de beide ingangen zijn en X de uitgang van elke NAND poort dan geld de waardheidstabel in figuur 1. Samen met figuur 2 is de werking van de viruswarner nu heel eenvoudig te volgen. Wanneer op pen 1, 2, 4 en 5 van IC 1 een laag signaal (0 volt) aanwezig is, dan is op de uitgangen 3 en 6 een hoog signaal (+5 volt) aanwezig. Op pen 9 en 10 is dan dus ook een hoog signaal aanwezig en op de uitgang, pen 8, staat dan volgens de waarheidstabel een laag signaal. Op de ingangen 12 en 13 is dus ook een laag signaal aanwezig met als gevolg dat op pen 11 een hoog signaal aanwezig is. Dit signaal schakelt de transistor in die op zijn beurt de zoemer aanstuurt.

De bouw

Het schema is klein genoeg om op een experimenteer printje te worden gebouwd. Denk eraan dat de zoemer goed is aangesloten (+ en -) en dat de zoemer bij een voedingsspanning van 5 volt werkt. De min van de schakeling wordt met pen 3,4,5,6 of 7 van de diskdrive connector verbonden. De plus wordt met pen 12 verbonden, de lijn TK0 met pen 15 en de lijn DKWE met pen 16. Indien men de diskdrive poort wil doorvoeren dan dienen alle lijnen te worden doorgevoerd. Vergeet de lijnen TK0 en DKWE niet!
De viruswarner kan naar eigen inzicht worden uitgevoerd. Wij geven hier een voorbeeld van een uitgebreide versie. Er kan een veel eenvoudiger versie worden gebouwd. Men kan bijvoorbeeld de beide leds weglaten of geen schakelaar inbouwen.

Benodigde componenten

* IC1	SN74LS00
* T1	BC547
* R1 t/m R3	330 Ohm
* D1	rode led
* D2	groene led
* S1	schakelaar, 1 maal om
* S2	drukknop, maakcontact
* BU1	zoemer, 5 volt gelijkspanning

° een 23 polige sub D male connector

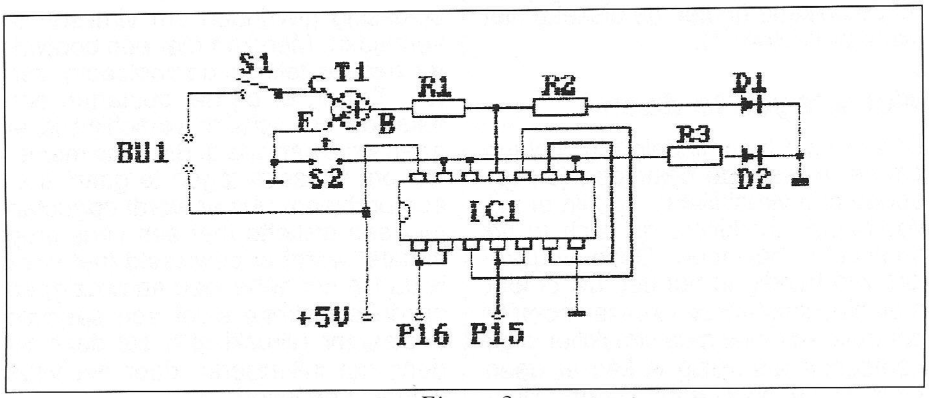
° (eventueel een 23 polige sub D female connector, alleen nodig indien de viruswarner doorgevoerd dient te worden)

Mogelijkheden

De schakelaar dient ervoor om de zoemer uit te zetten, dit is makkelijk met kopiëren omdat men anders met het kopiëren de zoemer hoort bij de eerste twee sectors.
De drukknop werkt alleen als de zoemer is aangezet en dient ervoor om te kijken of de viruswarner actief is, de leds werken dan wel (de groene gaat uit en de rode led gaat aan).
Wij hebben de viruswarner doorgevoerd omdat wij hem gebruiken op een Amiga 500, hier ziet men hem achter de computer zitten. Bij de Amiga 1000/2000 is dit niet nodig omdat men de viruswarner dan alleen hoort en niet ziet.

De software

Bij dit artikel hebben we een programma waarmee men de bootsector kan copiëren. Het programma maakt gebruik van het trackdisk.device en is in assembler geschreven. Dit is een library die functies bevat voor de diskdrive en het werken daarmee vergemakkelijkt. Een andere mogelijkheid om de diskdrive vanuit je eigen pro-



Figuur 3

gramma te gebruiken is met behulp van de hardware registers. Het programma is met behulp van de seka assembler geschreven. De listing moet eerst ingetikt worden en vervolgens met A [return] [return] te worden geassembleerd. Daarna kan men het programma opstarten met G [return] [return].

Mogelijkheden

Met de eerste functietoets (F1) leest men een bootblock in. Deze kan men met functietoets F2 weer wegschrijven (diskette eerst wisselen). Nu kan men deze nog een keer kopiëren door nog eens op functietoets F2 te drukken. Het programma kan worden beeindigd door op F3 te drukken.
De hardware was ontworpen door D.J. Brandt en de software door H. van der Pol.

De Seka listing:

```
;- Boot-copier             -
;- (c) Hans van der Pol -
;- Het programma maakt  -
;- gebruik van het          -
;- trackdisk.device        -
;- Er wordt op een een- -
;- voudige manier           -
;- duidelijk gemaakt hoe-
;- een copieer programma-
;- werkt...                  -

;- library offsets          -
;- exec.library             -
;- voor het werken met  -
;- de andere libraries  -
openlib        = -408
closelibrary  = -414
allocmem       = -198
freemem        = -210
opendevice     = -444
closedevice    = -450
doio            = -456

;- intuition.library  -
;- voor scherm opbouw -
openscreen     = -198
closescreen    = -66
openwindow     = -204
closewindow    = -72
printitext     = -216
drawborder     = -108

;- het begin adres van -
;- de exec.library      -
execbase        = 4

;- op dit adres lezen      -
;- we het toetsenbord uit -
toets           = $bfec01

;- nu begint het eigenlijke -
;- programma                   -

begin:

;- registers op stapel zetten -
 movem.l d0-d7/a0-a6,-(sp)

;- intuition.library openen -
 move.l execbase,a6
 lea intname,a1
 jsr openlib(a6)
 move.l d0,intbase

;- graphics.library openen -
 move.l execbase,a6
 lea gfxname,a1
 jsr openlib(a6)
 move.l d0,gfxbase

;- dos.library openen -
 move.l execbase,a6
 lea dosname,a1
 jsr openlib(a6)
 move.l d0,dosbase

;- een eigen scherm aanmaken -
 move.l intbase,a6
 lea screen_defs,a0
 jsr openscreen(a6)
 move.l d0,screenhd

;- in het eigen scherm een -
;- onzichtbaar window zetten -
 move.l intbase,a6
 lea window_defs,a0
 jsr openwindow(a6)
 move.l d0,windowhd
 move.l windowhd,a0
 move.l 50(a0),a0
 move.l a0,rastport

;- het menu regel voor regel -
;- afdrukken -
 lea text1,a1
 move.l #0,d0
 move.l #0,d1
 bsr print

 lea border1,a1
 move #0,d0
 move #0,d1
 move.l intbase,a6
 move.l rastport,a0
 jsr drawborder(a6)

;- dit is de loop waar in het -
;- programma eerst komt-
;- deze kan men eventueel zelf -
;- uitbreiden -
beginloop:
 cmp.b #$5f,toets
 beq boot_sector
 btst #6,$bfe001
 bne beginloop

;- tekst in eigen window -
;- afdrukken              -
print:
 move.l intbase,a6
 move.l rastport,a0
 jsr printitext(a6)
 rts

;- hier volgt de routine om de -
;- bootsector uit te le- zen... -
boot_sector:

;- eigen geheugen reserveren om-
;- daar later de bootsector in -
;- te zetten -
 move.l execbase,a6
 move.l #$2c00,d0   ;de lengte
 move.l #$10003,d1
 jsr allocmem(a6)

;- in diskbuffer staat waar het -
;- gereserveerde geheugen be- gint-
 move.l d0,diskbuffer

;- de msgport uit de wind- owhd   -
;- halen, voor het TD device -
 move.l windowhd,a0
 move.l 86(a0),msgport

;- het alom geroemde track- disk  -
;- device openen -
 move.l execbase,a6
 lea devicename,a0
 move.l #0,d0
 lea iorequest,a1
 clr.l d1
 jsr opendevice(a6)

;- hier wordt de tekst afge- drukt -
;- wat er gaande is -
lullaby:
 lea text10,a1
 move.l #0,d0
 move.l #0,d1
 bsr print

 lea text11,a1
 move.l #0,d0
 move.l #0,d1
 bsr print

;- de bootsector wordt inge- lezen -
 jsr td_read

;- motor van DF0: uitzetten -
 jsr td_motoroff

;- nieuwe status weergeven  -
 lea text10,a1
 move.l #0,d0
 move.l #0,d1
 bsr print

 lea text12,a1
 move.l #0,d0
 move.l #0,d1
 bsr print

;- wachten op de toets die toestemming -
;- geeft om te gaan schrij- ven       -
smash_key:
 cmp.b #$5d,toets
 bne smash_key

aiwa:
;- de ingelezen bootsector wegschrijven-
 jsr td_write
;- motor van DF0: uitzetten -
 jsr td_motoroff

;- de laatste gegevens door- geven -
 lea text10,a1
 move.l #0,d0
 move.l #0,d1
 bsr print

 lea text13,a1
```

```
 move.l #0,d0
 move.l #0,d1
 bsr print
 lea text10,a1
 move.l #0,d0
 move.l #0,d1
 bsr print
 lea text14,a1
 move.l #0,d0
 move.l #0,d1
 bsr print
;- nieuwe loop die vraagt of je wilt -
;- stoppen, lezen of nog es schrijven-
noges:
;- nog eens wegschrijven? -
 cmp.b #$5d,toets
 beq aiwa
;- nieuwe inlezen? -
 cmp.b #$5f,toets
 beq lullaby
;- stoppen? -
 cmp.b #$5b,toets
 bne noges
;- Er is nu 'stoppen' gekozen -
druit:
;- trackdisk device sluiten -
 move.l execbase,a6
 lea iorequest,a1
 jsr closedevice(a6)
;- geheugen teruggeven -
 move.l execbase,a6
 move.l #$2c00,d0
 move.l diskbuffer,a1
 jsr freemem(a6)
;- eigen window sluiten -
 move.l intbase,a6
 move.l windowhd,a0
 jsr closewindow(a6)
;- eigen screen sluiten -
 move.l intbase,a6
 move.l screenhd,a0
 jsr closescreen(a6)
;- gfx.library sluiten
 move.l execbase,a6
 move.l gfxbase,a1
 jsr closelibrary(a6)
;- dos.library sluiten -
 move.l execbase,a6
 move.l dosbase,a1
 jsr closelibrary(a6)
;- intuition.library sluiten -
 move.l execbase,a6
 move.l intbase,a1
 jsr closelibrary(a6)
;- de registers weer terug zetten -
;- op hun oude waardes -
 movem.l (sp)+,d0-d7/a0-a6
;- terug naar de CLI -
 rts
;- de bewuste sector inlezen -
td_read:
 move.l execbase,a6
 lea iorequest,a1
 move #2,28(a1)                ;2 = read
 move.l #$2c00,36(a1)          ;lengte=512*11*2
 move.l diskbuffer,40(a1)      ;buffer
 move.l #$0000,44(a1)          ;cylinder 0
 jsr doio(a6)                  ;uitvoeren
 rts
;- zelfde sector wegschrijven -
td_write:
 move.l execbase,a6
 lea iorequest,a1
 move #11,28(a1)               ;format (11)
 move.l #$2c00,36(a1)          ;lengte=512*11*2=$2c00
 move.l diskbuffer,40(a1)      ;pointer naar buffer
 move.l #$0000,44(a1)          ;cylinder 0 beginnen
 jsr doio(a6)                  ;uitvoeren
 rts
;- de disk drive motor uitzetten -
td_motoroff:
 move.l execbase,a6
 lea iorequest,a1
 move.l #0,36(a1)   ;lengte = 0 -->uit
 move #9,28(a1)
 jsr doio(a6)
 rts
;- hier staat de tabel met gegevens -
;- van ons zelfgemaakte screen -
even
screen_defs:
 dc.w 0,0           ;recht boven coord.
 dc.w 320,200       ;links beneden coord.
 dc.w 3             ;aantal bitplanes
 dc.b 0,0           ;kleuren
 dc.w 2             ;view
 dc.w 15            ;custom screen
 dc.l 0             ;geen andere font
 dc.l 0             ;titel (geen)
 dc.l 0             ;geen eigen gadgets
 dc.l 0             ;geen bitmap
;- dit zijn de gegevens van de zelfgemaakte -
;- window -
even
window_defs:
 dc.w 0,0           ;links boven coord.
 dc.w 320,200       ;rechts beneden coord.
 dc.b 1,3           ;kleuren (pen)
 dc.l $200          ;IDCMP flag
 dc.l 0             ;geen gadgets
 dc.l 0             ;geen eigen gadgets
 dc.l 0             ;checkmark
 dc.l 0             ;window naam
screenhd: dc.l 0 ;screen pointer
 dc.l 0             ;geen eigen bitmap
 dc.w 319,199       ;minimale coord.
 dc.w 320,200       ;maximale coord.
 dc.w 15            ;custom
;- de benodigde gegevens voor tekst -
;- staan hier -
even
text1:
 dc.b 3,0     ;kleuren
 dc.b 0       ;mode (4 = inverse)
even
 dc.w 65,10 ;coördinaten
 dc.l 0      ;standaard font
 dc.l txt1  ;tekst zelf
 dc.l text2 ;volgende regel
txt1:
 dc.b '* BOOTBLOCK COPIER *',0
even
text2:
 dc.b 3,0,0
even
 dc.w 65,25
 dc.l 0
 dc.l txt2
 dc.l text3
txt2:
 dc.b '(c) Hans van der Pol',0
even
text3:
 dc.b 3,0,0
even
 dc.w 20,55
 dc.l 0
 dc.l txt3
 dc.l text4
txt3:
 dc.b 'F1 - Read boot sector from df0:',0
even
text4:
 dc.b 3,0,0
even
 dc.w 20,75
 dc.l 0
 dc.l txt4
 dc.l text6
txt4:
 dc.b 'F2 - Write boot sector to df0:',0
even
text6:
 dc.b 3,0,0
even
 dc.w 20,155
 dc.l 0
 dc.l txt6
 dc.l text7
txt6: dc.b 'Here I give you my messages:',0
even
text7:
 dc.b 3,0,0
even
 dc.w 20,115
 dc.l 0
 dc.l txt7
 dc.l 0
txt7: dc.b 'F3 - Quit this powerfull tool',0
even
text10:
```

```
 dc.b 0,1,4
even
 dc.w 15,175
 dc.l 0
 dc.l txt10
 dc.l 0

txt10: dc.b '
                                ',0

even
text11:
 dc.b 3,0,0
even
 dc.w 17,175
 dc.l 0
 dc.l txt11
 dc.l 0

txt11: dc.b 'Reading boot-sec-
tor now!',0

even
text12:
 dc.b 3,0,0
even
 dc.w 17,175
 dc.l 0
 dc.l txt12
 dc.l 0

txt12: dc.b 'F2 to write
boot-sector!',0

even
text13:
 dc.b 3,0,0
even
 dc.w 17,175
 dc.l 0
 dc.l txt13
 dc.l 0

txt13: dc.b 'Copy finished',0

even
text14:
 dc.b 3,0,0
even
 dc.w 17,175
 dc.l 0
 dc.l txt14
 dc.l 0

txt14: dc.b 'F2 - Write sec-
tor again!',0

;- de gegevens voor de vier-
kant waarin -
;- de messages verschijnen  -
even
border1:
 dc.w 0,0       ;begin coord.
 dc.b 3,0       ;kleuren
 dc.b 0         ;mode (2 kan
ook)
 dc.b 5         ;aantal X,Y co-
ord.
 dc.l koord1    ;pointer naar
coord.
 dc.l 0         ;geen verdere
borders.

;- in koord1 staan de coordi-
naten voor -
;- de vierkant die op het
scherm verschijnt -
koord1:
 dc.w 10,165
 dc.w 270,165
 dc.w 270,190
 dc.w 10,190
 dc.w 10,165

;- de variabelen -
even
rastport:   dc.l 0
viewport:   dc.l 0
intbase:    dc.l 0
gfxbase:    dc.l 0
dosbase:    dc.l 0
windowhd:   dc.l 0
diskbuffer: dc.l 0

;- namen van gebruikte libra-
ries -
intname:    dc.b 'intui-
tion.library',0
gfxname:    dc.b 'gra-
phics.library',0
dosname:    dc.b 'dos.libra-
ry',0
devicename: dc.b 'track-
disk.device',0

;- trackdisk.device gegevens -
even
iorequest:
 dc.l 0,0
 dc.b 5,0
 dc.l mn_node

msgport:
 dc.l 0
 dc.w 0
 dc.l 0,0
 dc.w 0
 dc.b 0,0
 dc.l 0,0,0,0,0,0
mn_node: dc.b 'port',0
```

Zwetend zwoegt de ware computergebruiker zijn weg door de door zinderende zonnestralen overgoten dagen. Het zijn zware tijden mensen! Terwijl de medehuisgenoten genieten van het zonovergoten stralend zomerweer, zijn wij gedwongen ons heil binnenshuis te zoeken. En dit alleen maar omdat wij niet kapitaalkrachtig genoeg zijn om een NEC Ultralite LapTop en bijbehorend EtherLAN aan te schaffen!

Binnenin AmigaDOS (8)

Alweer de achtste aflevering. We naderen alweer met rasse schreden het einde van deze cursus. Deze keer nog de laatste restanten van de verbeterde AmigaDOS commando's.

De resterende commando's

Welke commando's zijn er nog overgebleven? We komen er nog zes tegen, te weten Run, Search, SetClock, SetDate, Statuse en Type.

Run

Run dient nog steeds om achtergrondprocessen om te starten. Wat is een achtergrondproces nu eigenlijk? De Amiga is een multitasking machine. Een dergelijke computer is in staat meerdere programma's (=taken of 'tasks') tegelijkertijd uit te voeren. Eigenlijk niet helemaal tegelijkertijd. De software in de Amiga zorgt ervoor dat de verschillende programma's op gezette tijden van de microprocessor gebruik kunnen maken. Deze wisseling van programma's gaat echter zo snel dat het lijkt alsof de programma's tegelijkertijd lopen. Het ene programma kan meer processortijd krijgen dan de andere. Dit is afhankelijk van hun belangrijkheid (lees 'priority'). Hoe hoger de priority, hoe meer processortijd deze programma's gemiddeld krijgen. Hoe start je nu vanuit DOS een programma op die (semi) tegelijk loopt met de andere al lopende programma's. Er zijn twee oplossingen.

1. Start een nieuw DOS venster op met NEWCLI of NEWSHELL en start het programma.

2. Start het programma met RUN.

NEWCLI en NEWSHELL hebben we een aantal afleveringen geleden al behandeld dus we volgen nu punt 2.

De syntax van RUN:

Run [programmanaam]

Wat is er veranderd in **Run**? Voorheen was het niet mogelijk het DOS venster te sluiten als er nog een, in dat venster, met 'Run' opgestart programma liep. Men moest dan het DOS venster tot minimale grootte terugbrengen en dat was dan dat. Het is nu echter mogelijk de uitvoer van 'Run' om te leiden, met het " teken weet u nog wel, naar een file. Als dat eenmaal gedaan is kan het venster gerust gesloten worden. Denk er dan wel om dat er geen invoer van DOS commando's meer mogelijk is. Het kan gebeuren dat het programma invoer eist van het venster. In dat geval kunt u het DOS venster niet sluiten.

Een tweede verandering. De commando's die uitgevoerd moeten worden door 'Run' mogen ook van de zogenaamde 'Resident list' komen. Deze lijst is een commandolijst welke de commando's omvat die in het geheugen opgeslagen zijn. Als er een commando op deze lijst vermeld staat, dan is deze in het geheugen aanwezig en wordt bij aanroep meteen uitgevoerd.

Een voorbeeld van Run:

Run df2:c/CygnusEd

Met deze commandoregel wordt de 'CygnusEd' editor opgestart. Deze editor staat in dit geval in directory 'c' op drive 'df2'. Het is dus mogelijk een volledige 'path' vermelding te geven naar waar het programma te vinden is.

Search

Engels voor zoeken. Het commando leest een programma door aan de hand van de opgegeven string. Het commando Search zal naar onze menig niet zo vaak gebruikt worden; echter als u op zoek bent naar een bepaald zoekwoord in één van uw tekstbestanden en u heeft uw tekstverwerker niet geladen dan is het mogelijk dit vanuit DOS met het **Search** commando te doen.

De syntax van Search:

Search [FROM] filenaam [SEARCH|string] [NONUM][QUIET][QUICK][FILE]

Ook hier zijn weer enkele keywords te vinden.

FROM: deze is eigenlijk overbodig, echter in scriptfiles staat het wel netjes. FROM geeft aan uit welke file gezocht moet worden. U mag ook rechtstreeks de filenaam vermelden zonder het FROM keyword.

SEARCH: als u dit keyword gebruikt dient u er achter het (de) gezochte woord(en) te vermelden. Ook dit keyword mag u overslaan en de gezochte string meteen achter de filenaam vermelden.

NONUM: Search zal nu de regelnummers niet afprinten als het de gezochte string gevonden heeft.

QUIET: Search zal nu geen uitvoer geven op het beeldscherm. Het nut van deze optie was ons niet geheel duidelijk!

QUICK: Search zal de uitvoer nu wat bezuinigd op het beeldscherm toveren, tenminste dat is wat er in de officiële handleiding staat. We zijn dus wat aan het experimenteren geweest. We hebben onderstaande routines eens uitgevoerd.

search file1 coloradjust.c Gadget QUICK
search file2 coloradjust.c Gadget
cmp file1 file2

Het resultaat hiervan was NIL, oftewel niemendal, niente! Er was geen verschil tussen de uitvoer van de eerste en de tweede. Voorlopig tasten wij dus nog even in het duister omtrent deze optie.

FILE: zonder vermelding van FROM en de door te zoeken file zal Search nu gaan fungeren als een soort 'Which' commando. Het gaat zoeken of de achter SEARCH vermelde string ook als programmanaam op schijf voorkomt.

Wat is er veranderd in Search?

Search heeft nu een extraatje voor de script files. Als **Search** niets vindt zal deze de WARN(=5) flag zetten. Als het wel wat gevonden heeft wordt een 0 teruggegeven. Met een IF/ENDIF constructie is het dan mogelijk een selectie procedure op te zetten.
Een voorbeeld:
Search MijnTekst DigiView
de uitvoer zou dan kunnen zijn:

15 met DigiView zijn verbluffende...
79 DIGIVIEW heeft een afzonderlijk...

SetDate

Met **SetDate** kunt u de datum en tijd opnieuw instellen. Voorheen was dit behoorlijk lastig. Je kreeg regelmatig de meest verwarrende boodschappen op het scherm, dus wat was de oplossing, een clock module aanschaffen of de tijd met Preferences instellen.
De syntax van SetDate:
SetDate [filenaam][datum][tijd]

Wat is er veranderd?

De input van SetDate is wat minder streng geworden. Het is nu mogelijk de uitvoer van Date te gebruiken als input voor SetDate. U kunt met SetDate de datum- en tijdvermelding van een file veranderen.
Een voorbeeld.
Setdate MijnBestand

Hiermee veranderd u de datum- en tijdvermelding van de file 'MijnBestand' naar de actuele datum en tijd.

Status

Weer een commando wat specifiek is ontwikkeld voor de multitasking eigenschap van de Amiga. U krijgt met Status een volledig overzicht van de lopende processen op de Amiga. Het is geen uitputtend overzicht doch het geeft enige informatie over de lopende programma's.
De syntax van Status:
Status [procesnummer] [FULL] [TCB][CLI|ALL][COMMAND]

Er is een aantal zaken veranderd. **Status** ondersteunt nu ook de negatieve prioriteiten. Wat is een negatieve prioriteit, zult u zich afvragen. De prioriteiten zijn eigenlijk subjectieve waarden. De programmeur heeft ze bepaald. De meeste processen (lees programma's) zullen niet echt belangrijk, noch echt onbelangrijk zijn. De waarden liggen ergens in het midden. Nu is het zo dat bij de Amiga de prioriteit waarde tussen de waarden -128 en +127 moet liggen. De meeste processen hebben een prioriteit 0, de middenwaarde. Het kan voorkomen dat er processen zijn van meer belang, ze krijgen bijvoorbeeld de waarde 5, en er zijn minder belangrijke processen. Dezen krijgen bijvoorbeeld de waarde -5. **Status** kon deze negatieve waarden niet afbeelden in de vorige WorkBench versie.
Status ondersteund ook een nieuw keyword. Dit is COMMAND. Het is hiermee mogelijk de procesnaam, dit kan een programmanaam, etcetera zijn, op te geven. Status kijkt dan of het betreffende proces aanwezig is. Als het gevonden is geeft **Status** een 0 terug. Als het niet aanwezig dan wordt wederom de WARN(=5) flag gezet.
De overige keywords:
FULL: geeft een volledig overzicht van de lopende processen. (zie figuur 1)
TCB: geeft hetzelfde overzicht als in figuur 1 alleen de laatste kolom is weggelaten.
ALL = CLI: de opties ALL en CLI zijn eender. Het procesnummer wordt afgebeeld en de laatste kolom uit figuur 1 wordt afgebeeld.
Een voorbeeld
status command=df2:c/CygnusEd
Status zal nu een cijfer afbeelden welke overeenkomt met het procesnummer van het gezochte programma, in dit geval 'CygnusEd'.

De output van 'status'

-Procesnr.	Stackgrootte	GlobalVector	Prioriteit	Omschrijving
-Process x:	stk xxxxx,	gv xxx,	pri x	Loaded as command:x

(de x'en zijn in werkelijkheid waarden of strings)

Figuur 1

Type

Het laatste DOS commando. Er is hier niet zoveel veranderd. **Type** dient nog steeds om programmafiles op het beeldscherm, printer of in filevorm tevoorschijn te toveren.
De syntax van Type:
Type [FROM] filenaam [TO naam] [OPT H| OPT N][HEX][NUMBER]

De opties zijn:

OPT H = HEX: Type beeldt nu de hexadecimale waarden van de file af. **OPT H** en **HEX** zijn hetzelfde.
OPT N = NUMBER: Type geeft nu voor elke regel een regelnummer mee. **OPT N** en **NUMBER** zijn eender.
De opties **FROM** en **TO** zijn optioneel, dat wil zeggen, het is niet nodig ze te gebruiken. U kunt ook meteen de filenaam vermelden achter Type.
U kunt een file 'redirecten' naar een andere file door gebruik te maken van het TO keyword en door hierachter de filenaam op te geven of door het 'redirection' teken " te gebruiken. Geef dan achter dit teken de filenaam op en het resultaat volgt.

What's next? (Wat volgt?)

We hebben tot nu toe alle WorkBench 1.2 en WorkBench 1.3 commando's besproken.
We hebben gezien hoe we een eigen DOS commando moeten maken en we hebben een idee gegeven van de diverse drivers en handlers. De MountList en de Startup-sequence zijn voorbij gekomen, en als laatste, we hebben de nieuwe SHELL omgeving besproken. Wat volgt er nu nog? We gaan even wat dieper in op de Preferences.

De Preferences

De preferences is een programma die u op de WorkBench 1.3 diskette kunt vinden in de 'prefs' directory. Op de WorkBench1.2 diskette staat het programma gewoon in de 'root' directory. Even een waarschuwing, wat we in dit gedeelte behandelen is grotendeels niet van toepassing op de WB 1.2 preferences.
De meeste Amiga bezitters gebruiken de Preferences wel eens. Al is het maar voor het opnieuw instellen van de kleuren, (Houd hiervoor de volgende C-Info in de gaten. Hierin zal waarschijnlijk een routine staan waarmee de kleuren van het actuele scherm in te stellen zijn) of om de baudrate van de seriële poort te veranderen. Hoe roepen we de Preferences op. Ten eerste door vanuit de WorkBench op het 'prefs' icon te drukken en vervolgens de 'preferences' icon te selecteren. Dit is echter niet de methode waarvoor we deze cursus hebben op-

gezet. We willen dit doen vanuit DOS. Hoe dan vraagt u zich af? Even een kleine syntax beschrijving:

Preferences [POINTER][PRINTER][SERIAL]

Aangezien de 'preferences' in de prefs directory staat hebben we de keuze om met behulp van het 'CD' commando de 'prefs' directory tot de actuele directory te maken of door een zogenaamde 'path' vermelding voor 'preferences' te zetten. In dit geval kiezen we voor de tweede oplossing. De aanroep luidt; **prefs/preferences**. Het kan gebeuren dat u de perferences op een andere schijf in drive 'df2' heeft staan. De aanroep verandert dan in **df2:prefs/preferences**.
We geven geen opties op omdat we in het hoofdmenu van de 'preferences' willen komen. Het zou kunnen gebeuren dat u alleen de printerinstellingen wilt veranderen. Het is dan met de vernieuwde 'preferences' mogelijk om meteen in het 'printermenu' te komen. Het enige wat u dient te doen is de optie PRINTER achter 'preferences' te vermelden. Deze mogelijkheid geldt ook voor POINTER en SERIAL waarmee u in respectievelijk het 'change pointer'- en 'change serial' menu terecht komt.
Eerst het hoofdmenu van de 'preferences'. Op het eerste gezicht lijkt er niets anders, schijn bedriegt echter. Het vroegere CLI ON/OFF icon is verdwenen. Dit was eigenlijk ook een onzinnige functie dus er niets aan de hand. Voor het overige zijn hier geen extra dingen bijgekomen dus we gaan naar het volgende menu: het 'change printer' menu. Het 'graphics select' gadget is vervangen door 'graphic 1' en 'graphic 2'. Het 'graphic 1' gadget staat voor dezelfde menukeuze als het vroegere 'graphic select'. Wat is er anders geworden in dit menu, of laten we het anders stellen, wat is er nieuw in dit menu? Eén ding, er is een nieuwe grijsgradatiekeuze bijgekomen genaamd 'Gray scale 2'. Deze gradatie is volgens Commodore bedoeld ter ondersteuning van de A2024 monitor welke een maximum van 4 grijstinten toelaat. Het gebruik van deze 'gray scale 2' levert dan ook een wat grover geprint plaatje op. Meer veranderingen zijn er niet in 'graphic 1'. Dit in tegenstelling tot 'graphic 2', immers dit menu bestond niet in de 'preferences' van WB 1.2.

Hoe ziet dit menu eruit?

Linksboven ziet u het **smooting** gadget. Deze dient voor het gladder laten verlopen van diagonale lijnen. Niets is voor niets echter, het gaat wel ten koste van de printsnelheid. Als u nu veel diagonale lijnen in uw printwerk heeft en dit een gladder aanzien wilt geven, moet u 'smooting' op ON zetten. Even een opmerking, het werkt alleen als u in de grafische mode print. Een tekstverwerkingspakket als 'Word Perfect' zal er dus niet mee werken. Een tweede opmerking. De het printer.device kan printen achteruit. Dit kan oplopen tot een minimum van 3172 kleuren! Voor de 'gewone' zwart-wit-printers heeft deze optie (vrijwel) geen betekenis.
Dithering. Alweer een optie voor de eigenaars van een kleureprinter. De term 'dithering' staat voor 'het op zodanige wijze printen van puntjes in verschillende kleuren dat deze voor het oog als één kleur zichtbaar wordt.' Dit is vooral van belang als uw printer maar een beperkt aantal kleuren printen kan. Er is keuze uit drie algorithmes, te weten:

De instellingen van 'density' (1-7)

Nr	CBM 1000	EpsonQ	EpsonX	EpsonXOld	HP-Laserjet
1.	120x72	90x180	120x72	60x72	75x75
2.	120x144	120x180	120x144	120x72	100x100
3.	240x72	180x180	240x72	120x72	150x150
4.	120x216	360x180	120x216	240x72	300x300
5.	240x144	360x180	240x144	120x72	300x300
6.	240x216	360x180	240x216	240x72	300x300
7.	240x216	360x180	240x216	240x72	300x300

Tabel 1

F-S 'dithering' werkt niet samen met de 'smoothing'. Automatisch zal dan de 'ordered' dithering worden gekozen.
Vervolgens gaan we één gadget naar rechts. het **left offset** gadget treffen we hier aan. Dit bepaald de linker kantlijn van de bladzij. De offset dient in inches in te worden gegeven. Echt Amerikaans! We hebben een omreken formule opgesteld:
inch = 0.393701 * centimeters

Rond het getal af op 1/10 achter de komma en vul dit getal in, nadat u het CENTER gadget op OFF gezet heeft. Het zal waarschijnlijk wel duidelijk zijn, dit **center** gadget dient om het plaatje in het midden te zetten. (Ze hebben duidelijk nog niet van Bose gehoord... grapje!)
Density. Hier is de keuze uit 1 tot 7. Hiermee bepaalt u de printdichtheid ofwel de grafische mode waarin de printer komt te staan. (zie tabel 1)
We gaan een rij lager en beginnen weer aan de linkerkant. **Color Correct**. Dit menupunt is alleen van belang voor de bezitters van kleurenprinters. Hiermee onstaat de mogelijkheid om de printerkleuren beter aan te laten sluiten bij de beeldschermkleuren. Onder 'R', 'G', 'B' ziet u staan, **Colors=4096**, vooropgesteld dat u geen van de gadgets R, G of B ingedrukt heeft. Door de kleurencorrectie gaat het aantal kleuren dat

Ordered: de printer gaat op een soort C64 wijze kleurtinten produceren. Door de kleuren met behulp van geordende bitpatronen te printen ontstaat het effect van meerdere kleuren.
Halftone: een manier welke veel lijkt op de methode die men toepast in kranten. De kleurenintensiteiten worden bereikt door 'halftone' te printen, dat wil zeggen, de aanslag van de printnaalden wordt verminderd waardoor er een lichtere tint ontstaat. Dit geeft het effect van meerdere kleuren.
F-S: kleur verdelingen onstaan hier door gebruik van de Floyd-Steinberg methode. Hiervoor heeft u wel een printer nodig die meer dan 150 DPI (dots per inch) kan printen.

We gaan weer een menupunt naar rechts. Dit is weer voor alle printerbezitters. **Scaling** heet het menupunt en bepaald de uiteindelijke grootte van het plaatje. Hier is keuze uit twee mogelijkheden:
Fraction: normale schaling wordt door het printer.device uitgevoerd.
Integer: elke dot op het scherm wordt hierdoor een even aantal keren geprint. Dit is afhankelijk van de in density gekozen resolutie. Bijvoorbeeld een plaatje van 320 dots breed kan 320, 640 of 960 dots worden op papier.

We gaan een regel lager kijken. Aan de linkerkant vinden we twee gadgets

De structuur van 'system-configuration'

Offset	Type	Omschrijving
0	BYTE	FontHeight
1	BYTE	PrinterPort
2	WORD	BaudRate
4	----	een structure TimeVal voor instelling van de toetsrepeteer snelheid.
12	----	TimeVal structure voor vertraging tussen indrukken toets en repeteren.
20	----	TimeVal structure voor instelling 'double click'.
28	WORD	Matrix van 36 WORDS waarin de vorm van de muispointer is opgeslagen.
100	BYTE	XOffset
101	BYTE	YOffset
102	WORD	Kleur 0 muispointer (color 17)
104	WORD	Kleur 1 muispointer (color 18)
106	WORD	Kleur 2 muispointer (color 19)
108	WORD	Pointerticks
110	WORD	Kleur 0 scherm
112	WORD	Kleur 1 scherm
114	WORD	Kleur 2 scherm
116	WORD	Kleur 3 scherm
118	BYTE	ViewXOffset
119	BYTE	ViewYOffset
120	WORD	ViewInitX
122	WORD	ViewInitY
124	WORD	EnableCLI (nvt in WB 1.3)
126	WORD	Printertype
128	BYTE	Matrix van 30 elementen waarin de naam van de geselecteerde printer is opgeslagen.
158	WORD	PrintPitch
160	WORD	PrintQuality
162	WORD	PrintSpacing
164	WORD	PrintLeftMargin
166	WORD	PrintRightMargin
168	WORD	PrintImage
170	WORD	PrintAspect
172	WORD	PrintShade
174	WORD	PrintThreshold
176	WORD	PaperSize
178	WORD	PaperLength
180	WORD	PaperType
182	BYTE	SerRWBits
183	BYTE	SerStopBuf
184	BYTE	LaceWB
185	BYTE	Matrix van 30 elementen waarin de WorkBench naam opgeslagen kan worden.
215	BYTE	sys_reserved1
216	BYTE	sys_reserved2
217	WORD	PrintFlags
219	WORD	PrintMaxWidth
221	WORD	PrintMaxHeight
223	BYTE	PrintDensity
224	BYTE	PrintXOffset
225	WORD	wb_Width
227	WORD	wb_Height
229	BYTE	wb_Depth
230	BYTE	ext_size

Tabel 2

die **Width Limit** en **Height Limit** zijn getiteld. Deze gadgets zijn invoer gadgets voor de hoogte- en breedtegrenzen van het plaatje. Om wat voor soort grenzen het gaat zien we in het volgende gadget.

Limits. Hiermee bepaalt u dus hoe de grenzen ingegeven worden. Er is weer een aantal keuzes:
Ignore: de grenzen worden niet door u bepaald.
Bounded: de grenzen geeft u op in de gadgets, Width limit en Height limit. De maximum waarden zijn voor Width limit, 4.0 inches en voor Height limit 5.0 inches.
Absolute: u geeft de hoogte en de breedte van het plaatje op in Width limit en Height limit. Dit zijn de twee absolute waarden voor het geprinte beeld. Het is mogelijk dat hierdoor een erg vervormt plaatje ontstaat aangezien het printer.device niet meer de dimensies aanpast.
Pixels: u geeft de absolute grootte van het plaatje in pixels en in Width limit en Height limit respectievelijk de breedte en hoogte van het plaatje in pixels op.
Multiply: het plaatje kan een aantal malen worden vermenigvuldigd. U geeft in Width limit het aantal malen op dat het plaatje in x-richting moet worden vermenigvuldigd en u geeft in Height limit het aantal malen op dat het plaatje in de y-richting moet worden vermenigvuldigd.
De enige twee gadgets die dan nog over zijn, zijn 'OK' en 'CANCEL'. De functie van deze twee laat zich echter wel raden.
De overige schermen van de Preferences, te weten 'Change serial' en 'Edit Pointer', zijn identiek gebleven. Er zijn geen functies bijgekomen noch verdwenen.

Waar blijven de gegevens?

We hebben dit al eens eerder behandeld. Toch past dit wel goed in het bestek van dit artikel, dus let's go!
De gegevens van de Preferences worden opgeslagen in een aparte file. Kopieert u deze file, dan Kopieert u meteen alle eigenschappen van de WorkBench zoals die voor die schijf gelden. Dit zijn onder andere, de kleuren, de muispointer, de printerinstellingen, etcetera.
Al deze gegevens staan vermeld in die ene file die de naam 'system-configuration' draagt. Deze file vindt u in de 'devs' directory. Kopieer die file met behulp van het **Copy** commando, of gebruik de speciaal voor dit doel geschreven routine **CopyPrefs**. Voor CopyPrefs is het wel van belang dat u in het virtuele device DEVS de 'path' vermelding opgeeft naar waar DOS de system-configuration file moet plaatsen.

De system-configuration in detail

In het volgende gaan we dieper op de system-configuration in. Het is niet verwonderlijk als het u een beetje gaat duizelen. Enige voorkennis van de taal C is wenselijk.
Hoe ziet de system-configuration er uit? We gaan een aantal punten bespreken. Bekijk eerst tabel 2 eens en lees daarna verder.
Het eerste onderdeel van de system-configuration. Offset 1 is een BYTE welke de fonthoogte beschrijft die u in het 'main' scherm in kunt stellen. Als u 60 karakters geselecteerd heeft, zal hier een 9 verschijnen. Heeft u 80 karakters geselecteerd dan staat hier een 8. Bij 80 karakters zal de Amiga het font 'topaz 8' gebruiken en bij 60 karakters zal 'topaz 9' hetzelfde lot ondergaan. Het tweede onderdeel dat we aantippen, is offset 102-108. Dit zijn de kleuren van de muispointer. Elke kleur is één WORD (=16 bits) groot. Dit WORD is onder te verdelen in 4 x 4 bits. Elk
nibble (=4 bits) beschrijft één kleurcomponent. Een illustratie:

```
b15                    b0
0000  0001  0110  0010  = 0x0164
      R     G     B
```

Deze drie nibbles passen binnen 2 BYTE's ofwel binnen één WORD. Door de kleurcomponenten in hexadecimale notatie op te schrijven, krijgt men meteen de kleurinstelling te zien. Weer even terug naar de voorgaande illustratie, de kleur bestaat hier uit 1 deel ROOD, 6 delen GROEN en 4 delen BLAUW. Dit zal een lekker kleurtje worden!
De offsets 110-118 beschrijven de 4 WorkBench schermkleuren. Ook hier worden de kleuren op dezelfde wijze genoteerd.
Als laatste, offset 126. Hierin is de printernaam vermeld. Heeft u een EpsonX printer geselecteerd dan zal deze naam hier staan.

Toegift

Om nu snel te weten te komen hoe de Preferences instellingen zijn hebben we een kleine routine geschreven die de 'system-configuration' vanaf het virtuele device DEVS inleest en enkele waarden afbeeldt op het beeldscherm. U kunt zelf het programma aanpassen als u enkele waarden wilt weten uit de 'system-configuration' aan de hand van tabel 2.

```
Instructies bij readconfig.
-------------------------------------------
Programmanaam        : readconfig
Programmeertaal      : C
Gebruikt programma   : Aztek C v3.6a
Opties               : cc readconfig.c +L
                        ln readconfig.o -lc32
Aanroep              : readconfig
-------------------------------------------
Demonstratie programma voor uitlezen 'system-configuration'
-------------------------------------------------------------
#include "functions.h"
#include "exec/types.h"
#include "exec/memory.h"
#include "libraries/dos.h"
#include "libraries/dosextens.h"
#define CVTW(a,b) ((UWORD)b<<8L | (UWORD)a)
extern struct FileHandle *Open();
main()
{
 struct FileHandle *fp;
 char *buffer;
 int length,i;
 if(!(fp=Open("DEVS:system-configuration",MODE_OLDFILE)))
   {
    printf("Helaas geen system-configuration aanwezig!\n");
    exit(5);
   }
 if(!(buffer=(char *)AllocMem(260,MEMF_PUBLIC|MEMF_CLEAR)))
   {
    printf("Te weinig geheugen!\n");
    exit(5);
   }
 length=Read(fp,buffer,231);
 Close(fp);
 printf("\n-------------Preferences
   instelling-------------\n");
 printf("Font hoogte  : %2d\n",*buffer);
 printf("\nScherm x-offset  : %3d   Scherm y-offset  :
   %3d\n",
        *(buffer+118),*(buffer+119));
 printf("\nMousePointer kleuren --------\n");
 printf("Kleur 0 : %4x kleur 1 : %4x kleur 2 : %4x\n",
        CVTW(*(buffer+103),*(buffer+102)),
        CVTW(*(buffer+105),*(buffer+104)),
        CVTW(*(buffer+107),*(buffer+106)));
 printf("\nScherm kleuren --------------\n");
 printf("Kleur 0 : %4x kleur 1 : %4x kleur 2 : %4x kleur 3
   : %4x\n",
        CVTW(*(buffer+111),*(buffer+110)),
        CVTW(*(buffer+113),*(buffer+112)),
        CVTW(*(buffer+115),*(buffer+114)),
        CVTW(*(buffer+117),*(buffer+116)));
 printf("\nPrinternaam -----------------\n");
 for(i=0;i<29;i++)
   printf("%c",*(buffer+126+i));
 printf("\n");
 printf("\nInterlace instelling (0=uit) :
   %2d\n",*(buffer+184));
 FreeMem(buffer,260);
}
```

Tot slot

Tot zover weer deze aflevering van onze DOS cursus. Dit keer weer eens wat meer afwisseling. De volgende aflevering wordt de laatste. We gaan nog een keer in op het aspect "hoe eigen DOS commando's" in elkaar te frutselen! Dit moet echter wel in C gebeuren, BASIC leent zich niet zo goed voor dit doel. Heeft u interesse hiervoor en uw C kennis is ruim beneden peil, wel, we zijn van plan enkele listings af te drukken. Deze zijn vast de moeite wel waard. So stay Amiga-minded en laat het mooie weer u niet naar het hoofd stijgen!

Johan & johan.

Op het eerste gezicht lijkt een Teletekst-decoder voor de Amiga een overbodige aankoop. De eerste de beste middenklasse-TV beschikt al over zo'n ingebouwde decoder,dus waarom er nog eentje voor de Commodore aanschaffen? Daar kunnen echter diverse redenen voor zijn. Bijvoorbeeld het opslaan en bewerken van Teletekst-pagina's tot .IFF-bestanden, het uitprinten daarvan en/of de kleurenmonitor van een veelzijdige TV-tuner te voorzien. In deze test bekijken we de mogelijkheden en prestaties van de Microtext Teletext Adaptor.

Microtext Teletext Adaptor

Teletekst en TV-tuning op de Amiga

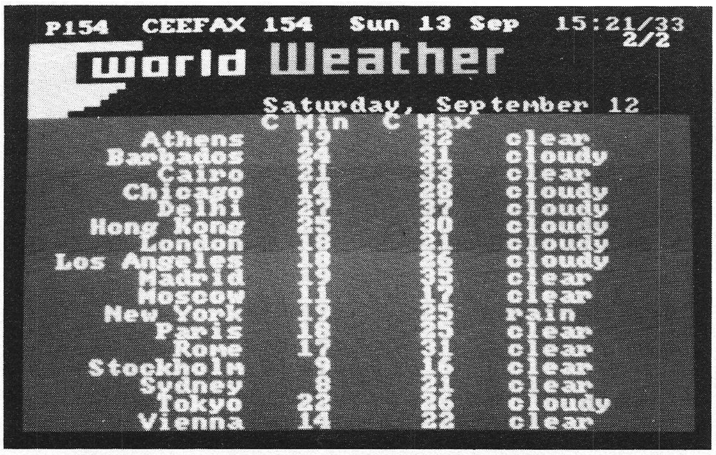
Teletekst is ook in Nederland een veel gebruikt informatiemedium. Waarschijnlijk omdat meer dan 50% van de huidige tv-toestellen met Teletext is uitgerust, wordt er ook veel gebruik van gemaakt. Bijvoorbeeld als aanvullende ondertiteling, het lezen van de weersverwachting en de vertrek/aankomsttijden. Een Amiga 500, 1000 of 2000 kan niet standaard met de kabel- en Teletekstsignalen uit de voeten. Daarvoor is een speciale Teletekst decoder annex tv-tuner nodig. Is deze eenmaal aangesloten, kunt u de Teletext-pagina's binnen de Amiga-hardware halen, met .IFF-compatibele software bewerken en naar een disk saven.

De adaptor-set

In Nederland heeft de Amiga-gebruiker de **internationale versie van de Microtext Teletext Adaptor** (N.B. *Telextext* is de Engelse benaming) nodig. Deze versie werkt in de meeste Europese landen met uitzondering van Engeland en Frankrijk.
De complete kit bestaat uit de volgende componenten:

- De **Microtext Teletext Adaptor**, een plat rechthoekig kastje met aan de voorzijde een keuzeschakelaar voor print en Teletext en aan de achterzijde een lintkabelconnector voor de parallelle poort, een voedingsaansluiting, een coaxiaal in-, een CVBS-uit-plug en een aansluiting voor de parallelle printer.
- Een **losse netvoeding** om die van de Amiga te ontlasten
- De **Microtext software**
- Een **DIN-RCA-kabeltje**
- Voor de Amiga 1000 is een speciale Male to Male 25-pins D-adaptor noodzakelijk
- Een **Engelse handleiding**

Het aansluiten is relatief eenvoudig. Pas alleen op met afwijkende parallelle printerpoort van het Amiga-model 1000. Alleen via een speciale 25-pins D-connector, male-to-male-type, anders gaat de handel kapot! Natuurlijk schakelt u de Amiga ook voor de montage uit.
Gelukkig gaat de printerpoort niet voor uw trouwe afdrukker verloren. De Teletext Decoder beschikt over een volwaardige parallelle printerpoort aan de achterzijde van het interface-modulatorkastje. Met het schakelaartje bepaalt de gebruiker de Teletext- of printstand.



Het weerbericht met Microtext

De installatie

De software maakt de installatie van de Teletext Decoder tot een sinecure. Gewoon de Amiga, printer en dan Teletext Decoder aanzetten, de software runnen en een keuzemenu voor de gangbare Europese tv-systemen verschijnt op het monitorscherm. Kies het juiste systeem en de Microtext Teletext Adaptor scant de tv-frequenties voor u. Worden er Teletext-data gevonden, dan komt daarvan de eerste regel op het scherm. Teletext vraagt daarop het gevonden kanaal te identificeren en het desbetreffende station een nummer te geven.
Het scannend zoeken stopt bij UHF-kanaal 68 waarop de software de tunerstanden naar de diskette wegschrijft. Alle informatie betreffende de tuning en de Default Language worden in het CONFIG_TELETEXT weggezet. Vervolgens kiest Microtext Teletext pagina 100 van kanaal 1.
Behalve voor de Amiga Teletext-ontvangst kunt u, zoals reeds vermeld werd, de decoder ook als tv-tuner voor de kleurenmonitor inzetten. Daartoe dient eerst het DIN-

RCA(tulp)-kabeltje te worden aangesloten. De DIN-plug gaat in het Teletext-interface. De twee tulppluggen gaan respectievelijk in de CVBS-video en de audio-in. Voor de weergave dient de Scartplug uit de monitor verwijderd te worden en schakelt u deze beeldbuis op composiet- video.

Het kiezen van Teletext-pagina's en -kanalen

Er zijn vier manieren om met de Microtext Teletext een bepaalde pagina op te halen. Drie van die manieren gaan met de muis en één met behulp van het toetsenbord. Wat die muis betreft is er keuze uit inklikken op + (linker muisknop) en -, via het indexnummer op het scherm en inklikken op de vier optionele pagina's uit de onderste Teletextlijn, de zgn. "Fasttext"-methode. Deze laatste methode is ideaal voor het snel doorbladeren van de textpagina's.
Het pagina kiezen met het toetsenbord gaat simpelweg door het intypen van het paginanummer. Dat paginanummer verschijnt linksboven in beeld en de bovenste regel kleurt tijdens het inladen groen.
Ook het kanaal kiezen is een makkie. De eerste vier zitten gewoon onder F1 t/m F4. De rest kunt via de Channel-optie uit het Select-menu bereiken.

Pagina-hulpjes

Om de Teletext-pagina's effectief te kunnen besturen beschikt de Microtext Teletext-software over een aantal nuttige **gadgets**. Deze hulpjes vindt u onder het Display-menu:

- Met **close** verlaat de gebruiker de Teletext-pagina's en keert hij/zij weer naar de Amiga Workbench terug.
- De + en - gaan een pagina voor- respectievelijk achteruit.
- **Reveal** laat de verborgen gedeelten, bijvoorbeeld de antwoorden van vragen of puzzels, zien.
- De **Review-optie** laat de laatste 16 in het RAM geladen pagina's zien. Ook handig voor het lezen van meerdere Teletext- pagina's achter elkaar.
- De **Hold** bevriest het scherm. Bijvoorbeeld van belang bij snel veranderende data.
- Met **ScreenToBack** stopt de Telext-display tijdelijk achter de andere zichtbare schermen. Van belang bij multi-tasking-toepassingen.
- De **ScreenToFront-optie** brengt de Teletext-display weer op de voorgrond.
- **Speak** laat de Amiga de hele pagina uitspreken. Jammer genoeg alleen in het Engels. Daar moest eens een Nederlandse versie voor de visueel gehandicapten aan toegevoegd worden. Deze spreekoptie staat als enige onder het Project-menu.

Verder staan onder het Display-menu de opties Default- en Set Language en Remove/Restore Gadgets.

Paginabewerkingen

In de eerste plaats gaat het bij de Microtext Teletext Decoder om het opslaan en weer inladen van de Teletext-pagina's. Er zijn onder het Project-menu twee formats beschikbaar:

- Een **compact format** voor het wegschrijven van ongeveer 800 Teletext-pagina's naar één standaard 3.5 inch diskette.
- Het **.IFF format** om latere (creatieve) bewerking met bijvoorbeeld DeLuxe Paint III mogelijk te maken.

De naamgeving van de weg te schrijven pagina's gaat automatisch op geleide van de paginanummering en -codering.
Wat uw TV niet (nog niet?) en de Amiga wel kan, is het afdrukken van Teletext-pagina's. In het zogenaamde **Fast format** wordt alleen de tekst, dus geen graphics, snel volgens de ASCII-code(s) afgedrukt. Dit format werkt ook op niet grafische printers en is de snelste methode om de Teletext-pagina op papier te krijgen.
Bij de **screendump** komen ook de graphics, de verschillende fonts en, indien afdruktechnisch mogelijk, ook de verschillende kleuren op het papier. Het afdrukken duurt relatief lang en bij snel updatende bladzijden is het van belang om eerst de Hold-modus te activeren. Anders kan het afdrukken een "soepzooitje" worden.
N.B.: De software neemt standaard aan dat er een Epson compatible printer aan de Amiga hangt. Bij andere emulaties de printerkeuze via de Preferences bijstellen.
De **language-opties** hebben uitsluitend betrekking op de karaktercodes van het desbetreffende land. Dat komt met name de weergave van de vreemde tekens (ë, ö, ä, ę e.d.) ten goede.
Met de **subcode-optie** kan de leverancier van het Teletext-signaal een extensie van vier digits aan het paginanummer toekennen. De mogelijkheden van de subcode hangen van het lokale gebruik af.

Programmeren

Iedereen een eigen Teletext met een eigen Amiga is het motto van Microtext. De software maakt het mogelijk om Teletext-pagina's vanuit elke programmeertaal aan te roepen, de .IFF formats in andere software te gebruiken en het informatie opzoeken vooraf te programmeren. Het automatiseren van de zoekprocedures biedt interessante perspectieven. Onder multi-tasking kan de software bepaalde gegevens voor een ander Amiga-programma opzoeken en alvast in het RAM of op schijf zitten. Daarbij valt te denken aan informatie over beursnoteringen, prijzen, vertrek en aankomsttijden, de weersverwachting, het sorteren van nieuwsberichten e.d. Onder het Project-menu staat het programma-submenu met:

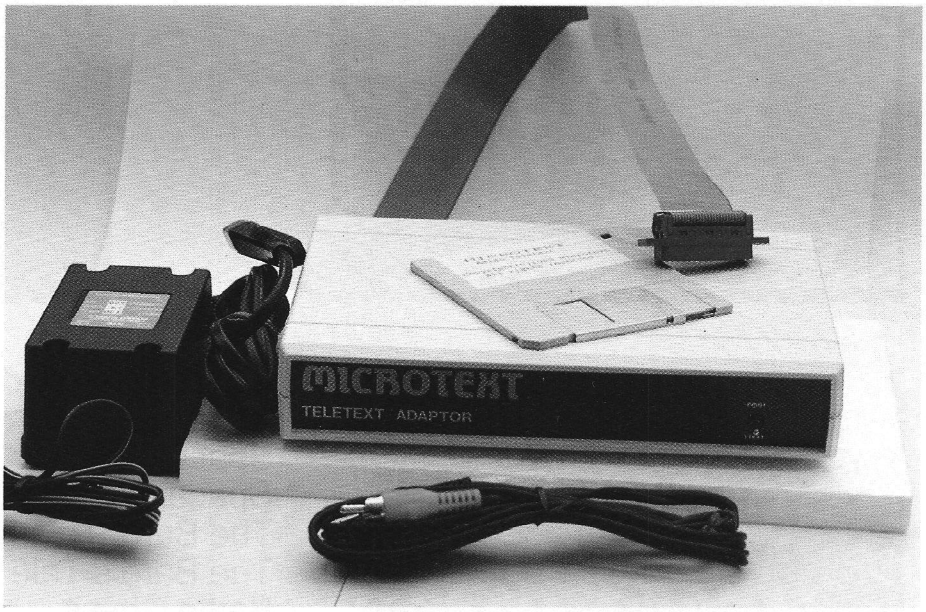
- **Selecting pages** op nummer, code of subcode
- **Delays**, het vasthouden van een actiesequentie in seconden, minuten en uren
- **Repeat**, het herhalen van een sequentie
- **ScreenToBack**, achtergrond-processing
- **Autorun** voor het maken van Teletext batch-files
- **Saves drawer** voor het specificeren van de output naar het desbetreffende bestand of subdirectory
- **Close Gadget**, voor het sluiten van het paginahulpje

Met behulp van de programmeeropties kunt u het Teletext Decoder-systeem geheel naar de eigen hand zetten en de vele creatieve mogelijkheden ontginnen.

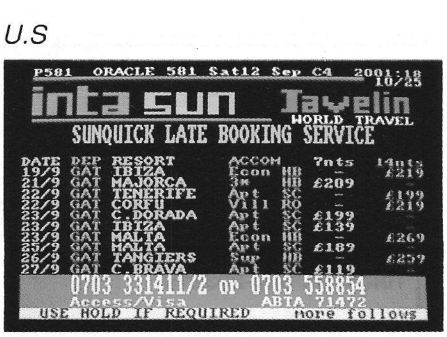
De praktijk

In de Amiga-praktijk van alledag komen twee belangrijke vragen naar voren: 1). "Hoe doet-ie het in kwalitatief opzicht?" En 2). "Wat heb je er eigenlijk aan?". *Vraag 1* valt bevredigend te beantwoorden. De kwaliteit van de Teletext-beelden en het bedieningsgemak mogen er best wezen. Ook de tunerfunctie voor de monitor doet het goed. Daarop verder geen aanmerkingen.
Vraag 2 valt aanzienlijk lastiger te beantwoorden. Het gaat immers om een apparaat van bijna *f* 800,- en dat zal niet iedereen er voor over hebben. Wie dan wel? Te denken valt aan de makers van slideshows en kabelkranten, Amiga-bezitters die Teletext-informatie in andere programmatuur willen gebruiken en voor het uitprinten of spreken van de Teletextpagina's. U zult zelf moeten uitmaken of de Microtext Teletext Decoder zijn prijs in het dagelijks gebruik waard is.
Voor alle Informatie: Cat & Korsch, Evertsenstraat 5, 2901 AK Capelle aan de IJssel.

U.S



SuperText met de Microtext Teletext Adaptor



Het digitaliseren van videobeelden op de Amiga is interessant voor grafische artiesten, reclamedoeleinden, picture databases, slideshows en creatieve videoprodukties. Vele Amiga-gebruikers zullen al over een videocamera of camrecorder beschikken en Digi-View van NewTek biedt hun de mogelijkheid om video-opnamen in .IFF- of RGB-bestanden weg te zetten. Daarna kunt u hen naar hartelust met .IFF compatible tekensoftware bewerken en/of in slideshows, videofilms, databases enz. opnemen.

Digi-View 3.0

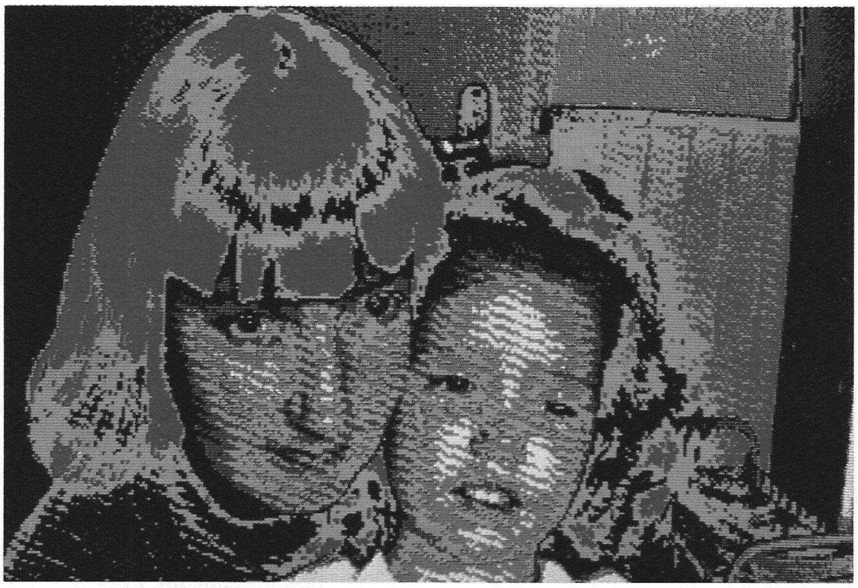
Video-digitizen in monochroom of kleur op de Amiga

Digi-View is volgens de makers van NewTek een combinatie van hard- en software voor de overdracht van kleuren- of monochrome opnamen van een videocamera naar de Amiga-computer. Zit het gedigitaliseerde beeld eenmaal in het Amiga-RAM, dan kan de gebruiker het als een grafisch (picture-)bestand saven en op schijf opslaan. Een aldus gesavede videoplaat kan vervolgens rechtstreeks in programma's of slideshows gebruikt, op de printer afgedrukt, via een modem verzonden of met teken- of videosoftware bewerkt worden. Het digitizen kan in monochroom of in kleur. Bij monochrome is een gewone zwartwit-videocamera voldoende. Dit soort camera's kunt u dikwijls ook bij de leverancier van Digi-View of als "bewakingssetje" kopen. Had u al een kleurenvideocamera, dan is ook deze geschikt. Het digitizen in kleur vergt verschillende opeenvolgende opnamen in Rood, Groen en Blauw (RGB) door het meegeleverde Digi-View-kleurenwiel.

De Digi-View-set

De PAL-versie voor **Digi-View 3.0** of **Digi-View Gold** bestaat uit de volgende componenten:

- Het **Digi-View videodigitizer-interface** is een keurig plat wit kastje ongeveer ter grootte van twee luciferdoosjes. Aan de voorzijde zit een standaard-RCA video-ingang. De Amiga-aansluiting is een parallelle connector. Het interface wordt verticaal, dus rechtopstaand, in de parallelle poort van de Amiga 500, 1000 of 2000 gestoken en neemt nauwelijks plaats in. **Let er altijd terdege op welke parallelaansluiting er op de Digi-View zit!**. Zoals bekend heeft de Amiga 1000 een ander type parallelle printerpoort dan de modellen 500 en 2000. Verwisseling beschadigt de computer. Heeft u een model dat niet op de parallelle poort in uw Amiga past, vraag dan bij aankoop een geschikte **Gender Changer** om tussen het digitizer-interface en de parallelle poort te zetten. De Digi-View doet het dan toch perfect en beschadigt de computer niet. Alleen steekt het geheel nu wel iets meer naar achteren uit.



Een gedigitaliseerd beeld

- De **Digiview digitizer-software** op slechts één diskette. Bij ons testexemplaar had een "halve gare" de 3.5 inch diskette met sterke hechttape in de doos geplakt. Zowel de aandrijfas als de sleufschijf zaten daardoor zeer vast aan het karton en onder de plaksel. Echt "bevordelijk" voor de

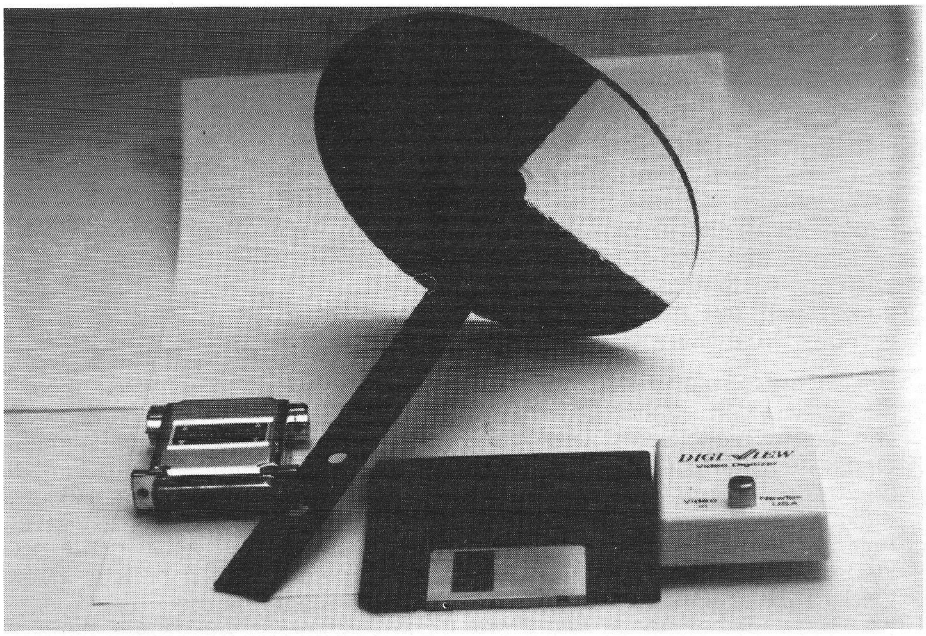
levensduur. Er zijn in deze toch betere oplossingen te bedenken.

- Het **kleurenwiel** met de vier kwarten Rood, Groen, Blauw en Helder. Er zit een bevestigingsbeugel met statiefaansluiting bij.
- Een duidelijke, 25 pagina's tellende **Engelse handleiding**

De Gender Changer en video-aansluitkabel zitten er niet standaard bij. Aanbevolen, maar niet echt noodzakelijk zijn een motor voor het filterwiel, een kopieerstand (een gewoon camerastatief voldoet ook, maar werkt minder handig) en een extra monitor met Y-adapter om de videocamera los van Digi-View te kunnen instellen. Bij een videocamera met een redelijk nauwkeurige tv-zoeker kunt u ook zonder die extra monitor aan de slag.

De hardware-installatie

Met alle goede componenten bij de hand is het aansluiten van Digi-View niet moeilijk. Natuurlijk dient alles wel uit te staan, want de Amiga blijkt soms een echt kasplantje bij het indrukken en uittrekken van pluggen en kabel. Gewoon de stroom er af en er kan niets engs gebeuren.
Het **Digi-View-interface** gaat zoals gezegd in de parallelle poort. Zit daar al een printerkabel in, dan zult u deze tijdelijk dienen te verwijderen. Nogmaals, let er op of een Gender Changer nodig is of niet. Vraag het aan de deskundige leverancier!
Het benodigde **videokabeltje** hangt van het gebruikte cameratype af. O.a. het merk Tegno Parts levert een heel scala aan plug-verloopkabeltjes voor videocamera's naar tulp RCA. De prijs ligt rond de *f* 40,-. Jammer is dat een aantal van deze kabeltjes in de praktijk behoorlijk teer blijken. Met name bij de camrecorder AV-uitgang. Wees dus voorzichtig bij het in- en uittrekken van de pluggen!
Daar het digitizen enige tijd (bij kleur zelfs drie maal zo lang) in beslag neemt, dient de opnameacmera stevig op het **statief** of op de **kopieerstandaard** geplaatst te worden. Dat voorkomt bewegingen en vermoeide armen.
Bij binnenopnamen kan een **set kopieerlampen** goed van pas komen. Door twee lampen met instelbare beugels aan weerzijde van het opname tableau te plaatsen, heeft men een egale verlichting met de beste opname-resultaten.
Het meegeleverde **filterwiel-beugeltje** is niet al te lang en dat kan bij groot uitgevallen videocamera's problemen



De Digi-View set

opleveren. Bij onze redactiecamrecorder van het VHS-C-type was het beugeltje lang genoeg. Of dat bij een full-size VHS-camrecorder ook past zal de praktijk leren. Een kleine zwart-wit-bewakingskamera gaf geen enkel montageprobleem. N.B.: de filters dienen zo dicht mogelijk voor de lens geplaatst te worden.
Het monteren van het **filterwiel zelf** is een kwestie van een plugje door de centrale wielas en het beugeltje te drukken. Voor het gebruik wel even de filters met een droge schone doek reinigen.
Voor het bekijken van de camera-scherpstelling en -uitsnede op de 1084 Commodore monitor van de Amiga kunt u of steeds de kabel tussen de Digi-View input en de CVBS-ingang van de monitor verwisselen of een Y-adapter en een extra kabeltje naar de 1084 of andere view-monitor gebruiken. Vergeet bij het bekijken op de Amiga monitor niet deze op composiet-video te zetten, anders (dus in de RGB-stand) komt er geen videocamerabeeld op de buis.
De afstelling van de monitor dient op geleide van de camera verlichting te gebeuren. Probeer door middel enkele experimenten te standaardiseren, zodat er beelden met steeds dezelfde helderheid, contrast en kleurenweergave worden opgenomen. Dat bespaart later veel retoucheerwerk op de Amiga.

De software-installatie

Na het laden van de Amiga Werkbank plaatst u de Digi-View-diskette in een diskdrive en activeert u diens ikoon met de muis. Het titelscherm biedt de keuze uit het oplossend vermogen van de te digitaliseren afbeelding (HI-RES, Interlace), Verticale en Horizontale Overscan en Color. Na instelling van deze preferenties gaat de gebruiker naar de overige menu's.
Er zijn drie hoofdmenu's:

- Het **Project-menu** behandelt de bestanden (laden en wegschrijven), het printen, het kleurenpalet, de histogrammen en het verlaten van het Digi-View-programma. Het opslaan van de gedigitaliseerde bestanden kan in Interchange File Format (**IFF**) of **RGB.** Moet het bestand in de toekomst nog verder bijgesteld worden, maak dan eerst een groot RGB-file. Voor verwerking in DeLuxe Paint, Aegis Animator en DeLuxe Video zijn de kleinere .IFF-bestanden aangewezen. Het te laden kleurenpalet beslist in hoeveel tinten Digi-View het picture-file opslaat. Houd daarbij rekening met het toekomstig gebruik. Kan het ontvangende programma bijvoorbeeld 32 of slechts 8 kleuren aan? Bij het afdrukken dient u eerst de Amiga uit te schakelen, het Digi-View-interface weer te verwijderen en de parallelle printerkabel in de poort te pluggen. Lastig, en daarom is het verstandig om alleen meerdere beelden

na elkaar (in plaats van stuk voor stuk) uit te printen. De histogrammens stellen de helderheid en het aantal pixels in.

- Het **Digitize-menu** biedt de keuze uit zwart-wit en in de drie opeenvolgende kleuren Rood, Groen en Blauw te digitaliseren. Er is ook een auto-optie voor de besturing (automatische RGB- opname) van het motor-aangedreven Digi-Droid-filterwiel.
- Het **Control-menu** omvat de beeldbewerking door Digi-View van de te digitaliseren beelden. Onder de kleuroptie staan de halfbright-(64 kleuren, 32 helder de andere 32 kleuren 50% minder helder) en HAM-modus (4096 kleuren onder Hold And Modify). Onder Palette maakt de gebruiker de keuze uit het aantal tinten of een bepaald (standaardpalet). In zwart-wit is er de keuze uit gewoon B&W en Line Art. Ook onder het control-menu staan de schuifregelaars voor de helderheid, kleurverzadiging, het contrast, de scherpte en de RGB-kleuren. De Dither regelt de korrel en beheerst samen met sharpness (scherpstelregelaar) de uiteindelijke scherpte-indruk. Een positief/negatief-schakelaar geeft de keuze tussen normale en negatieve weergave.
- Het **Camera-menu**, een submenu van Controls, behandelt de scantijd (langzaam en snel), de beeldgrootte (1/1, /2 en 1/4), de tracking en de breedte. Voor camrecorders en kleurenvideocamera's altijd de slow scan gebruiken voor de best mogelijke resultaten.

De genoemde vier menu's geven de Digi-View-gebruiker de volledige controle over het digitizing-proces, de kleur- en de scherpteweergave van de te saven beelden.

De praktijk

Met Digi-View moet je leren werken. Met behulp van het Engels handboekje lukt het al snel om een aardig plaatje te digitaliseren. Om echter optimale resultaten te kunnen bereiken is heel wat geëxperimenteer met kleur-, scherpte & dither-, camera- en save-instellingen nodig. Alles goed opschrijven dan heeft u er later geen omkijken meer naar.
Het digitizen zelf vergt enige handigheid. Met name bij de wat langer durende kleurenopnamen dient de camera als een huis te staan en het voorwerp niet te bewegen. Anders ontstaan onscherpe of bewogen beelden. Een goede verlichting is een must. Bij onvoldoende diffuus buitenlicht komen kopieerlampen voor een egale schaduwarme verlichting in aanmerking.
De mate van zichtbare korrel hangt van de gebruikte camera, de dithering en stand van de scherpteregelaar af. Minder scherpte geeft dikwijls een minder gekorrelde indruk. Bij het wegschrijven in minder dan 32 kleuren is het verstandig om een zo effen mogelijke achtergrond te kiezen. Dan blijft er tenminste nog wat kleuren voor het onderwerp over.
De zwart-wit-opnamen zien er voor de Amiga prima uit. Met de beeldscherpte-, contrast-, dither- en helderheidsregregelaars zijn de gedigitaliseerde goed op de eigen smaak af te stemmen. De printresultaten zijn zondermeer goed. Bij kleur staan of vallen de verkregen resultaten met de gekozen resolutie (mate van overscan), het kleurenpallet en het beschikbare vrije RAM. Vooral bij de lagere resoluties is het oplossend vermogen in pixels slechts geschikt voor creatieve korrelplaten of andere special effects. Alleen in de hoogste resoluties of interlace-modus valt het kleurenbeeld voor slideshow- en videotoepassingen te pruimen. Helaas vreten een hoog oplossend vermogen en een volledige overscan waarschijnlijk meer RAM dan u lief is. 512 KB was bij ons al snel te krap en met 1 MB ging het soms ook nauwelijks goed.
Het opname-instrument bij uitstek bleek de simpele zwart-wit-camera hoewel de kleurencamrecoder een goede tweede was. Bij gebruik van een gewone videorecorder (alleen geschikt voor monochroom digitaliseren) is een perfect stilstaand beeld onmisbaar. Eventueel kan ook een beeldplaatafspeler met een stilstaand-beeld-optie op het Digi-View-interface worden aangesloten.



Een gedigitaliseerd plaatje

Digi-View is een aardige (kleuren-)digitizer waarmee tal van effecten op de Amiga gemaakt kunnen worden. De hardware en software bieden alle mogelijkheden die de artiest, grafisch hobbyist, slideshow-maker en videoproducent zich in deze prijsklasse kunnen wensen. Bedenk wel dat het maken van goede gedigitaliseerde files flink wat werk en tijd kost. Met name dat gedoe met het filterwiel voor RGB-opnamen gaat allerminst snel. Een hulpmotor is al een vooruitgang, maar het blijft behelpen.
Digi-View 3.0 in de PAL-versie kost f 599,-. Een optionele Gender Changer rond de f 80,-. Voor info o.a.: Cat & Korsh International, Ervertsenstraat 5, 2901 AK Capelle a/d IJssel.

U.S


Charlie Chip's
SOFT WIR WAR
DOOR:
BUURMAN... PARTIJTJE TENNIS VOOR HET DONKER WORDT ?!
PFFT... DAT SOORT FLAUWE SPELLETJES DAAR BEN IK WEL OVERHEEN!
IK ONTWIKKEL ZELF SOFTWARE!
NEE... NIET OP DE COMPUTER!..
...OP DE TENNISBAAN!
NOU... VANDAAG NIET!..
IK HEB NOG 'N HOOP TE DOEN!
MAG IK EVEN BINNEN KIJKEN?!
IK HEB NAMELIJK OOK EEN CARAVAN!
ALLEMACHTIG!! DAT LIJKT WEL 'N HEEL NETWERK!
JA.. IK HEB ALLES BIJ DE HAND IN M'N EIGEN LAN A VAN!
..EN JIJ KOMT ZO LEKKER TOT RUST?!
NOU! REKEN MAAR...JE BENT ER EVEN HELEMAAL UIT HE?!
TJA... MAAR HET NEEMT NATUURLIJK WEL WAT RUIMTE IN!!
JA..DA'S WAAR!
...MAAR DAN HEB JE NATUURLIJK WEL ALLE COMFORT VAN THUIS!!
NOU JA... ALLE COMFORT!